Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE

Tomáš Hromádka

Using Artificial Neural Networks to Study Non-Linear Properties of Single Neurons in the Auditory Cortex

Katedra teoretické informatiky a matematické logiky

Vedoucí diplomové práce: RNDr. Věra Kůrková, DrSc.

Studijní program: Informatika, Teoretická informatika, Neprocedurální programování a umělá inteligence

I wish to express my gratitude to everyone who contributed to this work. Especially, I would like to thank Mike Wehr for generously providing whole-cell recordings of single neurons responding to natural sounds; and Hiroki Asari also for providing data, and help with STRF estimation.

I also thank Tony Zador for stimulating and thorough discussions, and, of course, Věra Kůrková, my advisor, for her constant support and interest during the past several years.

Prohlašuji, že jsem svou diplomovou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 14. srpna 2006 Tomáš Hromádka

Contents

Název práce: Studium nelineárních vlastností neuronů sluchové kůry pomocí umělých neuronových sítí
Autor: Tomáš Hromádka
Katedra: Katedra teoretické informatiky a matematické logiky
Vedoucí diplomové práce: RNDr. Věra Kůrková, DrSc., Ústav informatiky AV ČR
e-mail vedoucí: vera@cs.cas.cz
Abstrakt: Neurony v našich mozcích převádějí informace o okolním světě do elektrických akčních potenciálů. Klíčem k pochopení funkce neuronů, a poté neuronových sítí a mozku, je vědět, které části okolního světa jsou zastoupeny v aktivitě neuronů, a jakým způsobem je tato informace kódována. Tradiční metodou zkoumání funkce neuronu je zaznamenání odpovědi neuronu na podněty různé složitosti a následně snaha vysvětlit přenosovou funkci neuronu pomocí lineárního modelu. Většina neuronů však obsahuje nelineární přenosovou funkci. V této práci jsme použili umělé neuronové sítě ke studiu nelineárních přenosových funkcí neuronů sluchové kůry. Zaznamenali jsme podprahovou aktivitu neuronů jako odpověď na stimulaci přirozenými zvuky a poté jsme použili neuronové sítě v roli nelineárních aproximátorů k odhadu přenosových funkcí neuronů. Umělé neuronové sítě úspěšně aproximovali přenosové funkce a v průměru odhadli funkce neuronů nejméně stejně dobře jako lineární modely.
Klíčová slova: neuronové sítě, sluchová kůra, nelineární aproximace

Title: Using Artificial Neural Networks to Study Non-Linear Properties of Single Neurons in the Auditory Cortex
Author: Tomáš Hromádka
Department: Department of Theoretical Computer Science and Mathematical Logic
Supervisor: RNDr. Věra Kůrková, DrSc., Institute of Computer Science, AS CR
Supervisor's e-mail address: vera@cs.cas.cz
Abstract: Neurons in our brains continuously transform information about the external world into series of electrical impulses, or spikes. A crucial step in our understanding the function of neurons, and consequently neural circuits and the brain itself, is to know what features of our environment are represented in the activity of single neurons, and how these features are transformed into neuronal responses. Traditional approaches of studying neuronal function probe neurons with more-less complex stimuli, record neuronal responses, and fit a linear model trying to explain the stimulus-response relationship. However, most (if not all) neurons in the brain cortex display strong nonlinearities in their responses. Here we used artificial neural networks to study nonlinear stimulus-response relationships of single neurons in the auditory cortex. We probed neurons with a spectrally rich set of natural sounds and recorded their subthreshold activity. Then, we used neural networks as nonlinear approximators to obtain an unbiased estimate of stimulus-response functions of the neurons. We show that neural networks can recover the stimulus-response function of single neurons, and perform well when compared to standard linear models.
Keywords: neural networks, auditory cortex, non-linear approximation

Chapter 1

Introduction

The brain is one of the most complicated organized structures known to man. Composed of billions of cells (neurons) and orders of magnitudes more connections among those, the brain translates external inputs into internal percepts. The physical world around us is continuously being transformed into an internal representation; barrages of photons reaching our eyes, air pressure waves reaching our ears are translated into perception of an apple, or a nice music.

This—admittedly still mysterious—transformation is carried out by biological neural networks. Individual neurons inside these networks receive many inputs and compute their outputs, which are then propagated further in the brain. Obtaining a comprehensive description of these input-output transformations of single neurons is crucial for our understanding of information processing, sensory perception, and ultimately consciousness. Such complete description of the transformation should provide us not only with information about the type and range of the external stimuli (*i.e.*inputs) the particular neuron is sensitive to. Satisfactory description should also provide a successful prediction of neuronal responses to novel stimuli, *i.e.*stimuli which were not used to uncover neuronal responses in the first place.

Our understanding of the relationship between acoustic stimuli and neuronal responses in the auditory sensory area of the brain lags behind corresponding problem in, for example, the visual system. Indeed, despite many years of research we still lack a general consensus about what forms the basis of neuronal processing in the auditory area, *i.e.*what are the main acoustic features (simple sounds perhaps) represented by single neurons, and how are these features transformed by the neurons. And in most cases, even when we are able to approximate the

putative input-output transformation, these provide us with poor predictions of neuronal activity to different stimuli.

Neuronal properties become increasingly more and more complex as one follows the path of sound processing in the brain. Starting from the inner ear, in the early stages of processing, neurons respond well to simple sounds, such as pure tones. Moreover, responses of such neurons can be predicted reasonably well using a principle of linear superposition. That is, responses to complex sounds can be well approximated as a linear combination of responses to simpler sounds. As soon as one gets to the primary auditory cortex, *i.e.* the first processing stage in the brain cortex, most cells seem to have nonlinear properties, and their responses can no longer be predicted from responses to simple stimuli.

Multi-layer feed-forward neural networks are often used as nonparametric regression algorithm to approximate nonlinear functions. This use of neural networks requires no assumptions about the underlying function, which makes it very suitable for fitting nonlinear transfer function of neurons, especially in the auditory cortex, where the transfer functions might be complex. Although explicit models usually require fewer parameters, neural networks may guide search for novel features, which might be consequently incorporated into the explicit models.

Here we used multi-layer feed-forward neural networks to study (non)linear transfer functions of single auditory neurons responding to a rich repertoire of natural sounds. This study demonstrates that artificial neural networks can answer the question *what* are the main transformations in biological neural networks, despite failing to answer the question *how* these transformations are implemented.

Chapter 2

Background

Before presenting our main results it is necessary to summarize basic facts about artificial neural networks, [1] their history, motivations, structure and function. We primarily focus on artificial neural networks as a powerful tool for nonlinear function approximation. We mention general motivations, and describe several techniques that can be used for supervised learning in multi-layer feed-forward networks.

Main goal of this thesis is to study input-output transformations of biological neurons. We summarize the necessary facts known about such transformations performed by neurons in different sensory areas of the brain. As our main topic we will mention what is known about neurons in the primary auditory cortex, *i.e.*the first region of the brain cortex involved in processing sounds. We focus on what is known about the transformations these neurons perform, and the currently used techniques to recover these transformations.

2.1 Biological motivations

The brain is a powerful computational device, the function of which can be grossly oversimplified as depicted in Fig. 2.1. Thus we can view the brain as a black box, a mysterious device which receives external (sensory) stimuli from the environment, and then transforms these stimuli into actions, usually in the form of motor responses.

[1] We prefer to use the term *artificial* neural networks to distinguish theoretical neural networks from their biological counterparts

Figure 2.1: Schematic depiction of brain function. The brain, as shown in this figure, can be thought of as a "black box" receiving external sensory stimuli as input, and transforming the stimuli into motor actions, *i.e.*output.

Of course, the black box, *i.e.*the brain itself is structured. External sensory stimuli (light, sound, touch, *etc.*) stimulate their corresponding *sensory receptors* in sensory organs; *i.e.*photo-receptors in the eye retina, hair cells in the cochlea of the inner ear, and so on. The information from sensory receptors is then conveyed via so-called *subcortical structures* (brainstem, thalamus, *etc.*) into the brain *cortex*, a convoluted outer surface of the brain. The brain cortex itself can be subdivided into many *cortical areas* Fig. 2.2 having different functions and performing distinct computations.

In the brain cortex, information from sensory systems is first conveyed into so-called *primary sensory areas,* separately processing individual modalities. Thus visual information is first processed in the primary visual cortex, acoustic information in the primary auditory cortex, and so on. The primary sensory areas feed their outputs into *higher sensory areas,* multimodal areas, where information originating in separate sensory streams is combined, and the whole process culminates in *prefrontal cortex,* where the information is evaluated and decisions about consequent actions are made.

After making a decision the result is transformed via *motor cortex* into a motor response, an action. The process just described is a simplified version of an information flow in the brain. All cortical areas mentioned above are interconnected and all of them can receive input from, and also send their output to many other areas. Thus there is a continuous flow of information throughout the brain, during which the stimuli from external world are interpreted, evaluated, and transformed into actions.

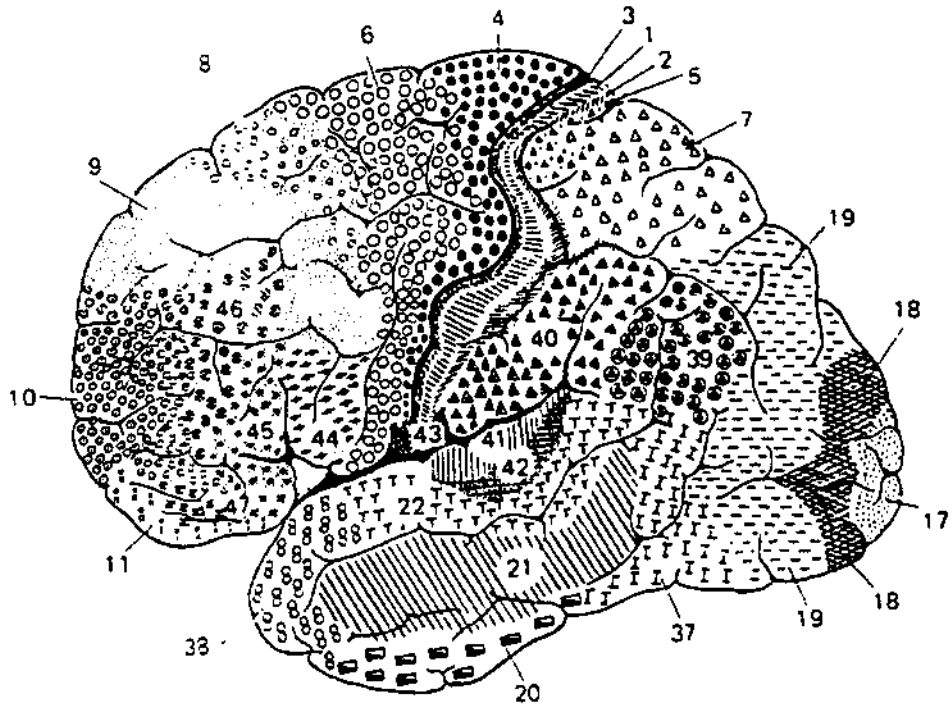

Figure 2.2: Brain cortex can be subdivided into many areas which perform various computations and are interconnected into a complicated network. Figure shows a view of lateral convexity of human brain cortex (adapted from Brodmann, 1909). For example, the primary visual cortex is area 17, and the auditory cortex encompasses areas 41 and 42.

What provides brain with immense computational power is its basic building and computational unit: a *neuron.* Neurons are nerve cells which come in a variety of shapes and sizes. A particularly prominent type of a neuron—*a pyramidal cell*—is shown in Fig. 2.3 and demonstrates the basic features of neurons.

Neurons consist of three morphologically and functionally distinct parts: cell body (*soma*), *dendrites,* and *axon.* (Kandel et al., 2000; Nicholls et al., 2001; Shepherd, 2004) Dendrites usually form an elaborate tree of branched processes emanating from soma. This tree forms the receptive (input) zone of the neuron, where signals from other neurons are registered, and transmitted to the soma. Signals are registered at *synapses,* which are junctions mediating inter-neuronal communication. At synapses the (originally electrical) signals are transformed into a chemical signal and then back into electrical fluctuations which propagate across the dendritic tree and, upon reaching the soma, are combined together and sent out in a form of brief electrical impulses called *action potentials,* or *spikes.* The transformation of (small) electrical fluctuations into strong and brief spikes takes place in an output zone of a neuron, and the spikes are then transmitted via neuron's transmission line called *axon* to other neurons. Although there are many types of neurons,

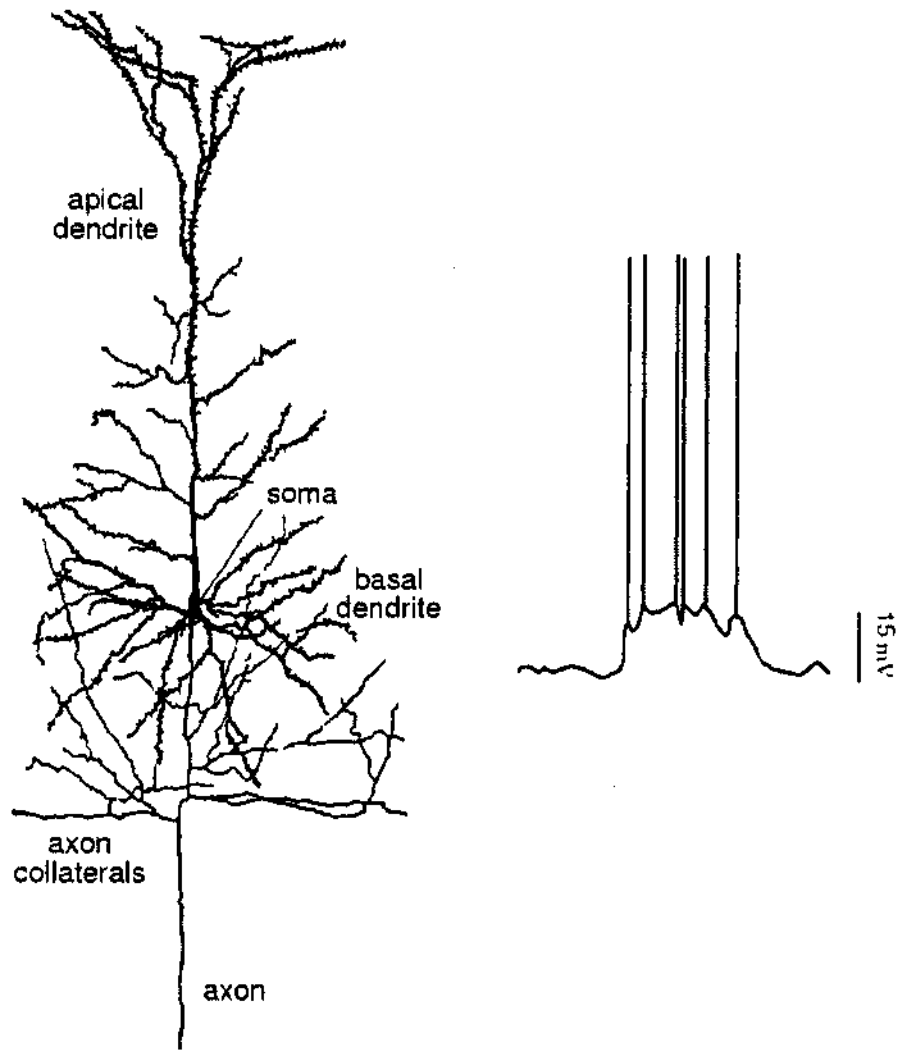

Figure 2.3: Pyramidal neuron and its activity. Left panel shows a typical neuron found in the brain cortex. The neuron consists of cell body (soma), receiving input from many dendrites, and outputting results of computations performed on inputs via axon. Right panel shows an example of neuronal activity. Subthreshold fluctuations are a result of combining inputs in the dendritic tree. When the subthreshold activity reaches threshold it results in a brief, all-or-nothing action potential (spike). The amplitude of subthreshold fluctuations and spikes is usually measured as voltage (in mV). (from Dayan and Abbott, 2001)

two very basic main classes can be recognized. Pyramidal cells mentioned above have long axons and primarily serve as long-range projections inside and between different cortical areas. Local projections inside groups of neurons are served by *interneurons,* which also contain elaborate dendritic tree, but their axons are usually short.

Neurons are densely packed inside brain cortex and it is estimated that there are approximately 10^{12} neurons in the brain cortex (Shepherd, 2004). In addition,

a typical neuron in the mammalian cortex receives thousands of synaptic inputs, and projects onto thousands of target neurons. Brain cortex contains an estimated 60.10^{12} neuronal connections and it is this extensive connectivity that forms a hallmark of neural circuitry. The brain, and brain cortex in particular can be then viewed as a massive computational device composed of (relatively) simple elements operating in parallel. It is this view that is conceptualized in the theory of artificial neural networks.

2.2 Artificial neural networks

The term artificial neural network has evolved to encompass a wide range of models and learning methods. In this section we will primarily focus on one of the classical types of neural networks: single hidden layer back-propagation network and its use as a nonlinear approximator. An interested reader can find *much more* detailed information, for example in Haykin (1999); Šíma and Neruda (1996).

2.2.1 Historical remarks

Artificial neural networks have been around for more than six decades. In 1941, Warren McCulloch, a neurophysiologist and psychiatrist, and Walter Pitts, a mathematician, decided to construct a simple model of neurons and their interconnections. Their joint effort was published in 1943 (McCulloch and Pitts, 1943) and remains cited as one of the first works that laid foundation for the field of neural networks. In their work, McCulloch and Pitts proposed an 'all-or-nothing' model of a neuron and showed that a network with a sufficient number of such neurons and properly set interconnections would, in principle, compute any computable function.

Another important contribution to the (at that time still very young) field of neural networks came from the psychology field with publication of *The Organization of Behavior* by Donald Hebb (Hebb, 1949). Hebb insightfully proposed a learning rule (now known as *Hebb's learning rule*) which states that effectiveness of a connection between two neurons is increased by the repeated activation of one neuron by the other across the connection.

Neural networks really captured imagination of scientists during 1950s with Rosenblatt's work on supervised learning and introduction of *a perceptron* (Rosenblatt,

1958). A few years later Widrow and Hoff formulated the concept of the *Adaline* (adaptive linear element), later expanded to Madaline (multiple adaline), with the main difference between perceptrons and (m)adaline lying in the training procedure (Widrow, 1962); (see also Haykin, 1999, for many other references). So, in the beginning of 1960s neural networks seemed to be destined to solve all possible problems of mankind.

However, in 1969 Minsky and Papert (Minsky and Papert, 1969) demonstrated fundamental limitations of what a single-layer perceptron could compute, and suggested there was no reason to assume that these limitations could be overcome by using multi-layer perceptrons. A decade of drought followed, research in neural networks stalled. Then Hopfield (Hopfield, 1982), and Rumelhart, Hinton, and Williams (Rumelhart et al., 1986) (among others) resurrected the field and brought it back to full glory. John Hopfield formulated computations performed by recurrent networks with symmetric connections and showed how these (dynamically stable) networks can store information. Rumelhardt et al. formulated the *back-propagation algorithm* for training of multi-layer networks, and demonstrated its power on key tasks. Consequently, back-propagation algorithm became the most popular algorithm for training of multi-layer perceptrons.

It must be noted that although the original motivation of the field of artificial neural networks was probably to model *and* explain function of biological neural networks in the brain, the reality is different. Indeed, starting with oversimplifications introduced by McCulloch and Pitts, the whole field deviated considerably from biological reality. On the other hand, the results and tools currently provided by artificial neural networks are phenomenally rich and inspiring. Thus although artificial neural networks might not tell us anything about *how* (*i.e.*what are the underlying connections and learning rules) a particular problem is solved by our brains, the networks can provide deep insights into *what* types of computations can be used to solve a particular problem faced by our nervous system. It must be noted, however, that recent advances in networks composed of spiking neurons (Maass and Bishop, 2001; Pavlásek and Hromádka, 2001), *i.e.*neurons resembling their biological counterparts, try to combine both the *how* and *what* approaches.

2.2.2 Structure and function

In the following sections we will continue our tour further into the world of neural networks with introduction of some basic terms, starting with *artificial neuron* (Sec. 2.2.3) as a basic building block of neural networks. We will describe

how neurons are combined into layers (Sec. 2.2.4), and on an example of a feed-forward network with single hidden layer we will show the basic computation performed by the network (Sec. 2.2.5). We will also touch upon basics of training of neural networks for particular tasks (Sec. 2.2.6), and mention several issues associated with training (Sec. 2.2.7).

2.2.3 Artificial neuron

Artificial neurons are the basic building blocks of neural networks. An example of artificial neuron (a *unit*) is shown in Fig. 2.4

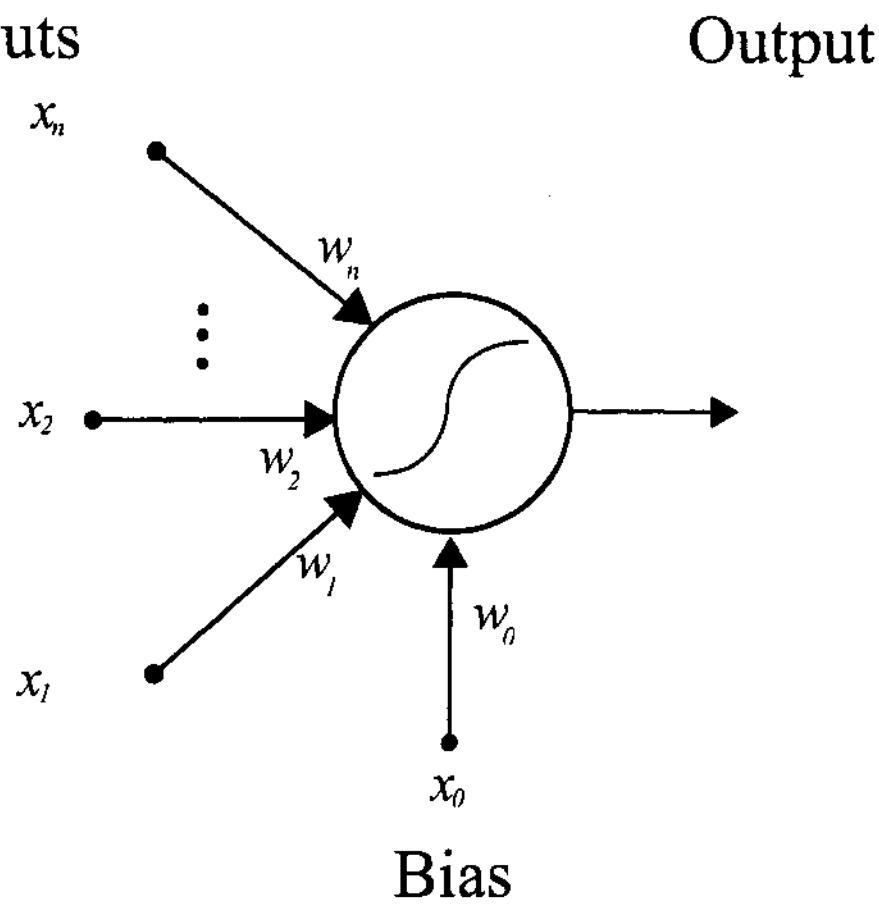

Figure 2.4: Artificial neurons as a basic building unit of neural networks. Each such neuron computes a weighted sum of its inputs (x_i), offsets the result by bias (w_0) and passes the result through activation function (sigmoid in this case) to obtain neuron's output.

Units are loosely based on properties of real biological neurons in a sense that they receive inputs, and output a specific transformation of these. Each unit computes a weighted sum of its inputs x_i to form its net activation ξ.

$$\xi = \sum_{i=0}^{n} x_i w_i. \tag{2.1}$$

where the subscript i indexes inputs, x_i denotes inputs, w_i denotes weights associated with each input, n stands for number of inputs. Note that the subscript i runs from 0. Here, and in the following text $x_0 = 1$, and w_0 denotes a *bias* of each unit, *i.e.*a value which offsets the inner product between the vector of inputs $(x_1, x_2, \ldots, x_n)$ and the vector of corresponding weights $(w_1, w_2, \ldots, w_n)$. The inputs x_i (connections) are sometimes called 'synapses' and their associated weights w_i 'synaptic weights,' in analogy with neurobiology.

Neuron (unit) then computes its output which is a (nonlinear) function of its activation (ξ), that is:

$$y = f(\xi). \tag{2.2}$$

The output y is then either passed as input onto another unit, or serves as a final result of the computation.

The function f which defines the output of the neuron is called the *activation function* and comes in many disguises. By far the most common activation function is the sigmoid function (Fig. 2.5), a strictly increasing s-shaped function that exhibits a "graceful balance between linear and nonlinear behavior" (Haykin, 1999). Throughout this text we will mostly use *hyberbolic tangent function* as our activation function, *i.e.* $f(\xi) = \tanh(\xi)$.

2.2.4 Network architecture

The way neurons are combined together defines the structure of a neural network: *network architecture.* In general there are two main types of network architectures: feed-forward and recurrent networks.

In *feed-forward networks* (Fig. 2.6) neurons are organized into *layers*, and outputs of neurons from one layer are fed onto inputs of neurons in the next layer. The example network in Fig. 2.6 is a multi-layer feed-forward network with a single hidden layer, or more specifically: two-layer feed-forward network. For the purpose of this text we do not consider the input layer a 'proper layer,' because no

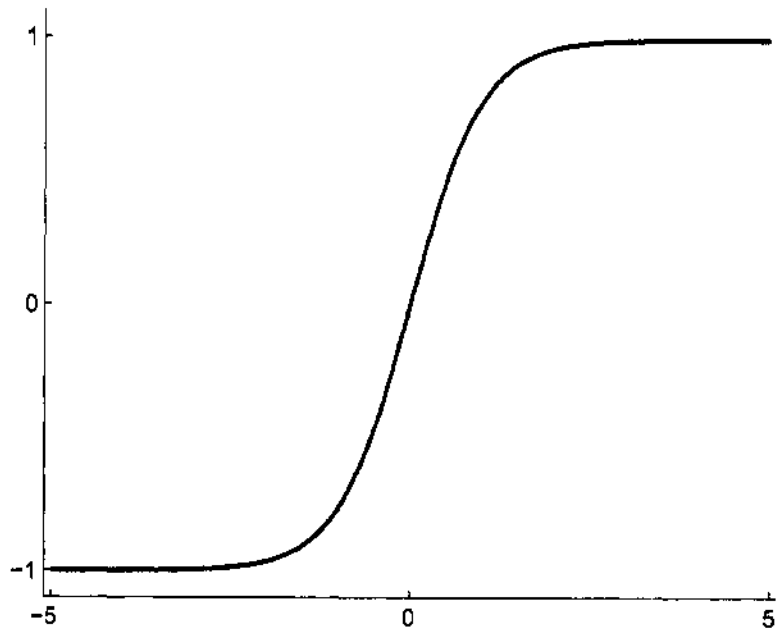

Figure 2.5: Sigmoid activation function of an artificial neuron. For the purpose of this text we use hyperbolic tangent (as shown) as activation function, unless stated otherwise.

computations take place in the input layer. Two-layer feed-forward network will be the main network architecture on which we will focus in the following text. For completeness however, we also mention the *recurrent networks* which distinguish themselves from feed-forward networks in that they contain at least one feedback loop.

The exact architecture of a network, *i.e.*number of inputs, number of layers, and number of units in each layer, depends on the problem the network is supposed to model, or solve. There is no known automated procedure to determine the optimal network structure for a given problem.

2.2.5 Network operation

Neural networks have two primary modes of operation: feed-forward processing and learning. In this section we describe the feed-forward operation of a feed-forward network with a single hidden layer and one output unit. We will touch upon network learning in the following sections.

Feed forward operation consists of presenting input values to the input units and passing the signals through the network in order to yield output from the output unit(s). Specifically, if we consider our example network from Fig. 2.6, we can describe computation of each hidden unit as (compare also with Eqs. 2.1–2.2):

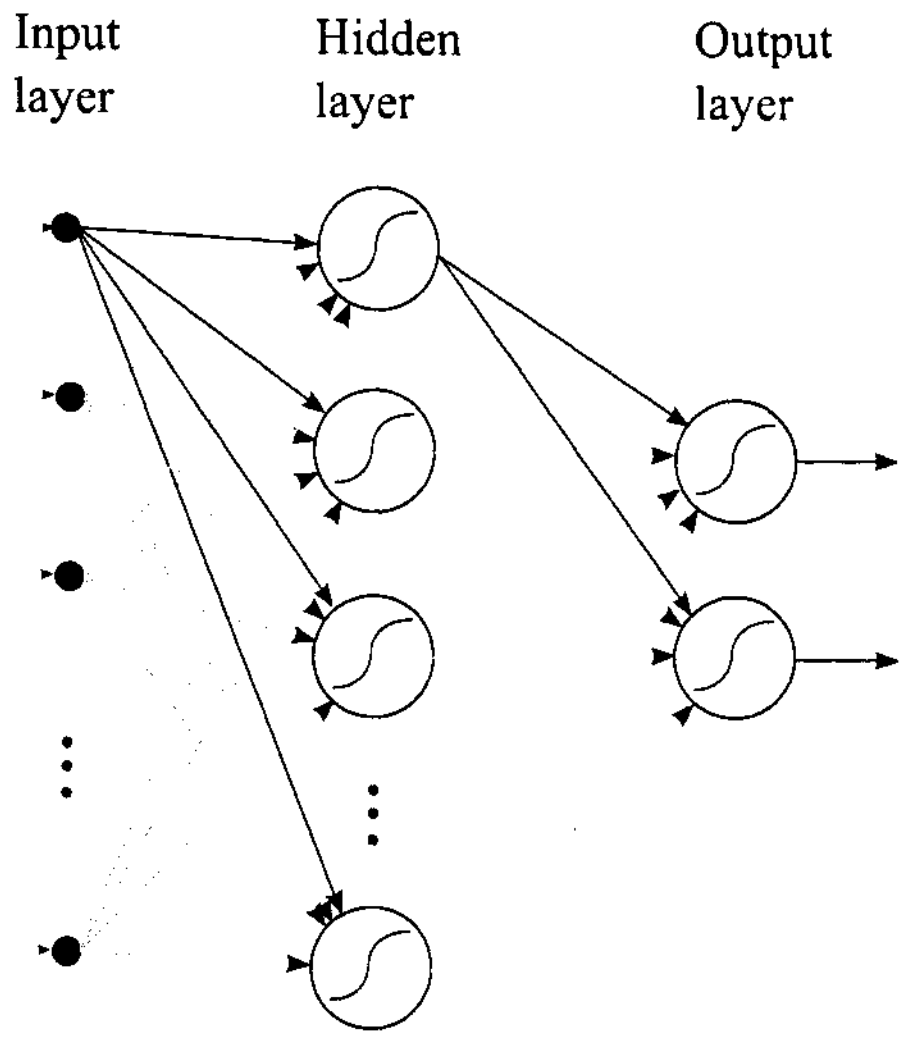

Figure 2.6: Architecture of a feed-forward neural network with one hidden layer. Input units ('Input layer') feed their outputs onto units in the hidden layer, outputs of which consequently feed onto output units ('Output layer'). Arrows show direction of 'information flow' in the network.

$$y_j = f_h(\xi_j) = f_h\left(\sum_{i=0}^{n} x_i w_{ji}\right), \tag{2.3}$$

where y_j is the output of the j^{th} hidden unit, computed as a function f_h of its net activation ξ_j. The net activation is computed as a weighted sum of inputs x_i, each weighted by corresponding (synaptic) weight w_{ji}. Here we consider total of n inputs for each hidden unit. By convention the first subscript (j) of weights indexes units on the "output side" of the expression, and the second subscript indexes units on the input side. Thus w_{ji} denotes the weight associated with i^{th} input coming to j^{th} hidden unit. As usual w_{j0} is the unit's bias, and the associated input is set to $x_0 = 1$.

The output unit then computes the its output representing the total output value z of the entire network as:

$$z = f_o\left(\sum_{j=0}^{m} y_j w_j\right), \tag{2.4}$$

where the output z is again computed as a function f_o of the net activation of the output unit. The net activation of the output unit is yet again weighted sum of inputs, in this case m inputs y_j representing the outputs of the hidden units. Each y_j has an associated weight w_j, with $y_0 = 1$ associated with the output bias w_0. In a general case we would consider more output units z_k, and in that case (according to our convention) the weight associated with the connection between k^{th} output unit and j^{th} hidden unit would be w_{kj}.

2.2.6 Network learning

Neural networks can learn. By learning we mean that a network created with certain parameters (weights, number of layers, number of hidden units, *etc.*) can adjust those parameters such that—when presented with certain input—produces the desired output. A network can learn using *supervised learning* which consists of presenting a training set of know input data and changing the network parameters to bring the actual outputs closer to the desired target values; or *unsupervised learning* which uses a task-independent measure of quality of the learned representation, *i.e.*there are no target values.

The network parameters are adjusted during *training,* which is a procedure whereby network is actually adjusted to do a particular job. The two basic training protocols are: stochastic and batch. In *stochastic training* the input patterns are chosen randomly from the training set, and the network weights are updated for each input presentation. In *batch training* all input patterns are presented to the network before learning takes place.

During learning, the parameters of neural network are adjusted according to a *learning algorithm.* Each learning algorithm has two basic requirements: it needs an estimate of *training error, i.e.*the algorithm needs to know how well the current set of parameters represents the desired output. The objective of the algorithm is then to minimize this error. An example of training error estimate is the *mean square error* between the target output and the network output:

$$E(\mathbf{w}) = \frac{1}{d}\sum_{k=1}^{d}(z(\mathbf{w}, k) - t(k))^2, \tag{2.5}$$

where E is the training error as a function of network weights $\mathbf{w}$, $t(k)$ is the k^{th} training input, $z(k)$ is the corresponding actual output of the network, with d elements in the training data set.

The learning algorithm also needs a *learning rule, i.e.*it needs to know how to adjust the parameters (weights) to improve network performance given the training error. An example learning rule uses the negative of the gradient of $E(\mathbf{w})$. The following equation shows how—in each iteration of the batch training process—the weights are adjusted proportional to the negative of the gradient of $E(\mathbf{w})$:

$$\Delta w_{ji}^{(t)} = -\varepsilon \frac{\delta E}{\delta w_{ji}}(\mathbf{w}^{(t-1)}), \tag{2.6}$$

where t is the current training iteration, and $0 < \varepsilon < 1$ is the *learning rate*, which determines how much the gradient influences weight adjustment. Algorithms using such learning rule then implement (a variant of) *gradient descent* on the (typically multidimensional) error surface, *i.e.*the algorithm is 'trying' to find a (global) minimum error.

Probably the most popular learning algorithm is the *backpropagation algorithm* (Haykin, 1999; Rumelhart et al., 1986; Šíma and Neruda, 1996). Standard backpropagation algorithm is a gradient descent algorithm in which the network weights are adjusted along the negative of the gradient of the training error. The term backpropagation refers to the manner in which the gradient is computed for multilayer networks with nonlinear activation functions. To compute the gradient from Eq. 2.6 the backpropagation algorithm takes advantage of the fact that all activation functions in the network are differentiable. For each unit, the gradient can be computed using partial gradients from the previous layer (thus the term backpropagation of errors). The power of the algorithm is that it allows us to calculate an effective error for each hidden unit, and thus estimate the proper values for the input-to-hidden weights. There are countless variants of the basic algorithm available, and the ones used in this text will be mentioned in more details in the *Methods* and *Results* sections.

2.2.7 Learning issues

Determining the correct network topology for a given problem can be hard. Whereas the number of inputs and outputs is given by the input and output data sets, the total number of weights or parameters in the network is not—or at least not directly. A naive application of backpropagation algorithm can lead to slow convergence, large training error, *etc.* There are number of heuristics available which improve network performance, speed up learning. Most of these are rather practical suggestions than strict mathematically proven solutions. We will mention some possibilities here, and the actually implemented heuristics will be mentioned in more detail in the *Methods* and *Results* sections.

Large differences in magnitude of individual elements of the input or output data sets are undesirable. During training, the inputs with values much higher than the rest of the input data set will drive the weight adjustment much more strongly that the small input values. To prevent such an uneven weight adjustment, it is recommended that both the input and output *data sets be standardized* before training. One possibility is to rescale the data sets, so that the mean value over the training set is zero, and the variance of the data set is the same, for example 1.

If the input data set is high-dimensional, for example in image recognition, where each image can be described by hundreds of pixels, one can employ some *data reduction technique,* such as *principal component analysis* (Dayan and Abbott, 2001; Hastie et al., 2001) to present only the most relevant dimensions and consequently speed up the training process and possibly also explore only the most relevant portion of the error space.

One of the most dreaded dangers when training neural networks is the danger of *overfitting.* Networks with large number of parameters can achieve very small training error because such networks have high expressive power and become 'tuned' to the particular training set. Nevertheless, the error on previously unseen data (test error) is unacceptably high, which means that the network failed to generalize properly. However, with too few parameters (too few hidden units, for example), the network simply might not have enough free parameters to fit the training set reasonably well and the test error is high as well. On the other hand, many free parameters offer higher-dimensional error space which in turn offers more ways to reach a minimum, for example. Thus we see that selecting and adjusting complexity of the network can be a deep problem in practice.

Several heuristics offer different solutions to prevent overfitting. Possibly the simplest one is to test the quality of generalization on *validation data set,* which is

a part of input data set *not* included in the training set. If the error on validation data set (validation error) is reasonably low, we can expect the network to generalize well. If, however, during training the validation error increases, the training should be stopped.

During the training process one can also change the number of free parameters by addition, or removal of hidden units *pruning*. In case of pruning the weights with very low values, *i.e.*those that contribute only a little to the final solution, are removed often together with their associated units. where the training success is determined also by error on previously topology adjustment - pruning, addition of new units

Regularization is another method for improving generalization. This method involves modifying the performance function, which in our case is the square mean error, and thus includes only contribution of network output. By adding terms which include contributions of network weights and biases into the performance function, such function will cause the network to have (for example) smaller weights and biases and the network output will tend to be smoother and less likely to overfit.

2.2.8 Neural networks as universal approximators

Intuitively, nonlinear multilayer networks—that is the ones with input layer, hidden units, and output units—seem to have greater computational or expressive power than similar networks that otherwise lack hidden units, or do not contain the nonlinearities. It seems that the mere presence of nonlinearities could enable the network to achieve any decision boundary and enable it to map virtually any input to any output.

Indeed, work by Kolmogorov and others (Kolmogorov, 1957; Kůrková, 1991, 1992) has shown that a network with a hidden layer, with sufficient number of hidden units, proper nonlinearities and weights can approximate just about any continuous function from input to output. Specifically, Kolmogorov proved that any continuous function $g(\mathbf{x}$ defined on the unit hypercube $I^n(I = [0,1]$ and $n \geq 2)$ can be represented in the form (Duda et al., 2004):

$$g(\mathbf{x}) = \sum_{j=1}^{2n+1} \Xi_j \left(\sum_{j=1}^{d} \psi_{ij}(x_i) \right), \tag{2.7}$$

for *properly chosen* functions Ξ_j and ψ_{ij}. Any input region can be scaled to lie in a hypercube, so this condition is not limiting. Recall the computation of a feed-forward multi-layer network as specified in Eqs. 2.3–2.4. We can interpret Eq. 2.7 in neural network terminology: let's assume we have a network with d inputs, $2n + 1$ hidden units, and one output unit. The net activation of each hidden unit is a sum of d nonlinear functions on input, one function per input unit x_i. Net activations are passed through nonlinear function Ξ_j, and the output unit sums the contributions of all hidden units.

In practice, Kolmogorov's theorem can be awkward for at least two reasons. The functions Ξ_j and ψ_{ij} can be extremely complex and far from nice, smooth non-linearities favored by neural networks and required by algorithms implementing gradient descent. The theorem also tells us little about how to find such functions based on the supplied data. Also other proof techniques were used to prove the universal approximation property of perceptron networks, for a survey see, e.g., Kůrková (2002); Pinkus (1998).

The fact that nonlinear feed-forward networks with hidden units can function as universal approximators enables us to use them to recover (unknown) nonlinear transformations from input to output data and study them. And that is the key power of multilayer neural networks, that they admit fairly simple algorithms where the form of the nonlinear transformation can be learned from training data.

2.3 Neuronal representations

A crucial step towards our understanding the function of neural circuits, and ultimately the brain, is to know the functions of individual neurons in the circuit. We need to understand what stimulus features are represented in the neuronal output (Barlow, 1972), and how those features are transformed to the output. This problem is usually called the *neuronal encoding* problem (Theunissen et al., 2004). Here we focus on neuronal encoding in the auditory system, the sensory system of the brain which processes sounds. In particular, we are interested in encoding performed by single neurons in the primary auditory cortex, the first place where information about sound reaches brain cortex.

2.3.1 Neuronal encoding

Neuronal encoding is the process by which sensory stimuli are transformed into neuronal activity in the corresponding region of the brain. For example, sounds

presented at the ear are decomposed into their frequency components in the inner ear, and—via subcortical structures—the information about sounds is relayed to the primary auditory cortex, used as input to single neurons and encoded (transformed) by these neurons. Neuronal activity is usually analyzed as sequences of action potentials (see Sec. 2.1 for details), or 'spike trains.' With sophisticated recording techniques, however, the neuronal activity can also be analyzed at the subthreshold level, *i.e.*before the activity is transformed into spikes, but after the neuron received its input.

Sounds are presented at the ear, and, as a response, neurons produce output. Different sounds evoke different outputs, and even the same sound usually evokes different outputs in different neurons. Thus to understand neuronal encoding we must be able to answer two closely related questions: *how* sounds are transformed into spike trains, and *what* aspects of sounds are important for producing spike trains (neuronal output).

In case of the visual cortex, it is widely agreed that oriented edges are the basic features of images represented by single neurons (Hubel and Wiesel, 1977). However, similar consensus is lacking in the case of audition (DeWeese et al., 2005), and such corresponding decomposition of sounds into simple components remains uncertain.

2.3.2 Analysis of responses to simple sounds

The traditional approach to study sensory neurons is to probe them with small sets of simple stimuli, designed to explore certain aspects of their response properties. Thus, properties of neurons in the visual cortex can be studied with oriented edges (Hubel and Wiesel, 1977). Auditory neurons are traditionally studied with pure tones of different frequencies (Ehret, 1997; Moshitch et al., 2006; Sally and Kelly, 1988). Perhaps the main reason for this approach is the fact the frequency components of sounds see to be important, given that the main function of the very first step in auditory processing is to decompose sounds into frequency components. This analysis produces *tuning curve* as the basic description of neuronal properties (Fig. 2.7).

Such simple stimuli proved valuable probing properties of neurons in very early stages of subcortical processing. However, many (if not all) neurons in the brain cortex seem to be driven by (unknown) complex features, such as vocalizations (Tian et al., 2001). Simple stimuli do not contain complex features and so are ineffective to test cortical neurons (Hromádka et al., submitted).

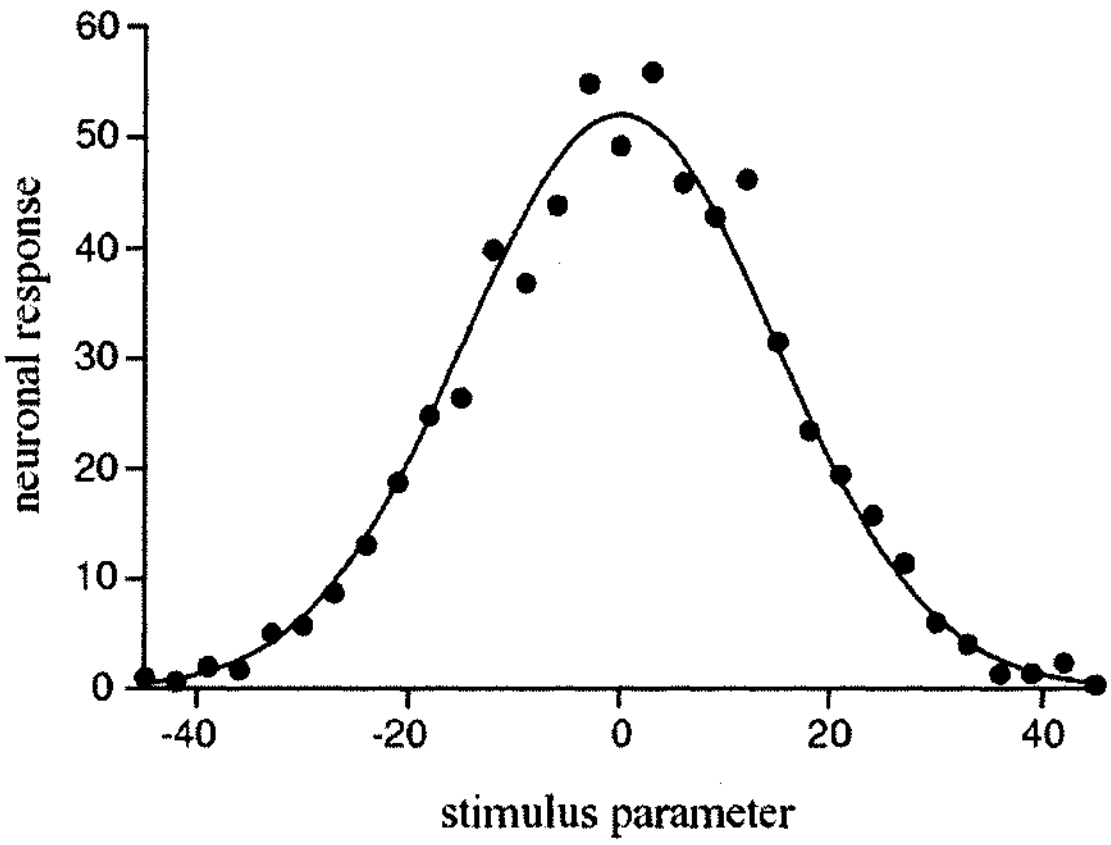

Figure 2.7: Tuning curve of a sensory neuron. Shown is magnitude of response of a single neuron probed with a simple stimulus, where one free parameter of the stimulus varied along the abscissa. For example, in case of single neuron in the primary visual cortex, the parameter would correspond to an orientation angle of a bar of light. In case of a single neuron in the primary auditory cortex, the stimulus parameter could correspond to frequency of pure tones (at fixed intensity and duration). In both cases the response could correspond to number of spikes elicited during the stimulus presentation. Note that, the shape of the tuning curve can vary. (adapted from Dayan and Abbott, 2001)

Most of cortical neurons (again, if not all) exhibit highly nonlinear stimulus-response relationships, even when probed with simple stimuli (Nelken et al., 1994). Thus it is not possible to use responses to a small set of stimuli to predict the response to any possible stimulus.

One way to circumvent these problems is to relax the assumptions about what the 'neuronal expectations' are and use *complex sounds* (stimuli) to test neurons. With this approach one expects the neurons to 'pick-up' the relevant features from the presented stream of stimuli. By complex stimuli we mean stimuli with many free parameters, and many possible interactions among the parameters; such as white noise, or so-called natural sounds like animal vocalizations, rustling of leaves, *etc.*

2.3.3 Linear analysis of responses to complex sounds

Reverse correlation, or 'spike-triggered average' (Dayan and Abbott, 2001) is the simplest method for analyzing responses of spiking neurons to complex *random* stimuli (Fig. 2.8). The complex stimuli are random, therefore unbiased, and this method provides the best linear fit of the stimulus-response transformation of the neuron. The fit is obtained by averaging portions of the stimulus which preceded spikes, with the underlying notion that those portions of the stimulus contained the features which were 'important' for the neuron, and therefore evoked a spike. For auditory neurons, the best linear fit is called the spectro-temporal receptive field (STRF) (Aertsen and Johannesma, 1981). STRFs have been estimated with a wide range of random stimuli, such as random chord stimuli (Blake and Merzenich, 2002; deCharms et al., 1998; Linden et al., 2003), and dynamic ripples (Klein et al., 2000; Kowalski et al., 1996).

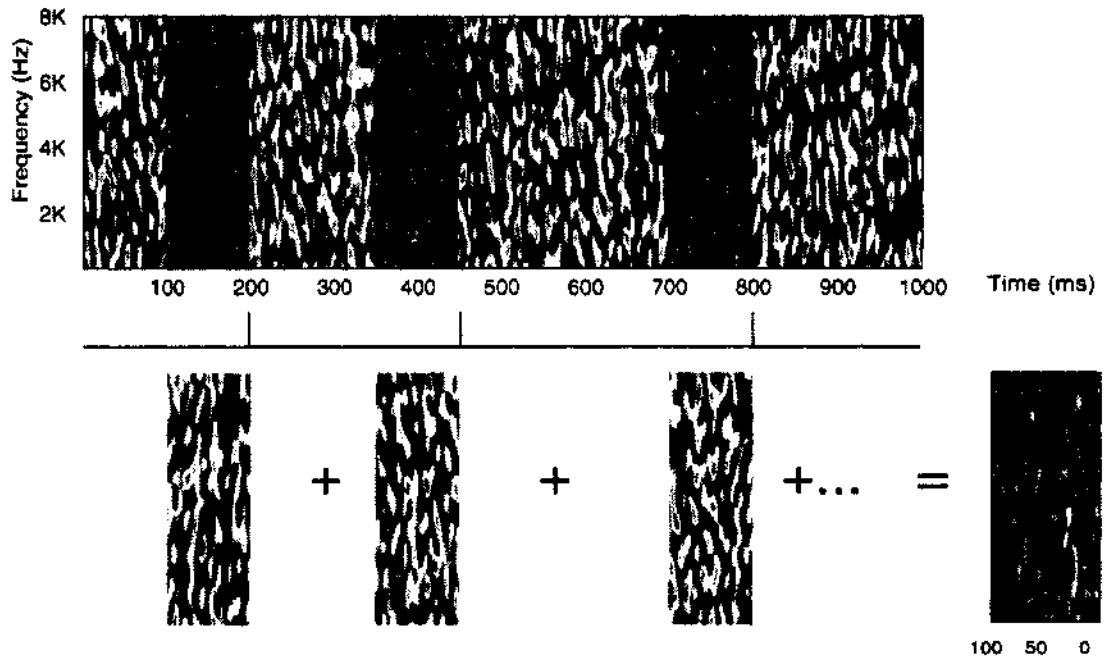

Figure 2.8: Estimating spectro-temporal receptive field (STRF) using reverse correlation. The STRF is estimated using spike-triggered average when the stimulus has white-noise characteristics. Every time a spike occurs, the stimuli that preceded it (100 ms before in this example) are averaged with stimuli that preceded other spikes. Eventually, the stimuli that are consistently correlated with changes in spiking build up, and other stimuli average out to zero. Here the stimulus is the spectrogram of a white noise sound (top). The STRF (bottom right) shows the sound frequencies that reliably occurred at a certain time lag before increases in spiking (approx. 15 ms in this case). (From Theunissen et al., 2004)

Natural sounds are another type of complex stimuli used for probing neuronal responses (Machens et al., 2004; Rotman et al., 2001; Sen et al., 2001). The motivation for using natural sounds as stimuli comes from the idea of *efficient coding* (Barlow, 1972; Nelken et al., 1999), which assumes that since sensory systems have evolved in natural environment, they might have evolved to efficiently code natural stimuli.

Big advantage of STRF analysis is that the estimated STRFs are linear fits. Therefore, STRFs provide a simple and straightforward description of neuronal properties. However, big disadvantage of STRF analysis is that the estimated STRFs are linear fits, and many of cortical neurons display nonlinear response properties. This fact is reflected in generally poor ability of STRF based models (especially STRFs based on natural sounds) to predict responses to new stimuli (Machens et al., 2004).

2.3.4 Nonlinear analysis of responses to complex sounds

One possible extension of the linear techniques is incorporation of second-order nonlinearity, which can be studied, for example, by performing the second-order Volterra kernel analysis of responses to natural sounds (Rotman et al., 2001). The second order kernel, however, usually consists of a large two dimensional matrix of parameters, which can be difficult to interpret in terms of encoding mechanism. Estimation of second-order kernel also usually requires much more data to collect for the analysis.

An alternative technique is to use artificial neural networks to learn (approximate) stimulus-response relationship of sensory neurons. Indeed a pyramidal neuron can be very well approximated as a feed-forward network with one hidden-layer and one output unit (Poirazi et al., 2003a,b). This approach requires no assumptions about the expected nonlinearity, and can be easily applied to analyze responses to natural stimuli.

Neural network approach in the *visual system* reveals most of the known and expected properties of neurons, and usually provides better prediction success (Einhäuser et al., 2002; Lau et al., 2002; Lehky and Sejnowski, 1990; Lehky et al., 1992; Prenger et al., 2004). The only auditory study (Bankes and Margoliash, 1993) was performed in the auditory thalamus and not cortex. Studying nonlinear response properties of auditory neurons probed with natural sounds can possibly reveal the possible computational subunits of the neuron, and lead to better prediction of its activity. Moreover, given that important acoustic features driving

neurons are not known, the hidden weights of neural networks trained on natural *images* correspond to oriented edges, *i.e.*the important visual features, it is not unreasonable to assume that neural networks trained on natural *sounds* might reveal the important acoustic features as well.

2.4 Background summary

Although the primary motivation for studying neural networks was the challenge imposed by complexity of human brain, neural networks have developed into an exciting world of their own, not longer related to the underlying biological motivation. Indeed, the powerful tools offered be neural network theory do not provide any insight into *how* a particular computation of transformation might be implemented in the real brain. These tools however can provide answers to questions *what* type of transformation is implemented in a particular (biological) system. For our purposes artificial neural networks provide an analytical alternative to conventional techniques which are often limited by strict assumptions of normality, linearity, variable independence, *etc.*

One of the classical problems in studying brain is the problem of neuronal encoding, whereby one tries to uncover what stimuli and how can excite a given neuron. Limitations of many techniques to study neuronal coding include insufficient stimuli, unable to drive neurons properly, and assumption of linearity of neuronal responses. Our main goal is to use neural networks to study nonlinear stimulus-response transformations in single auditory neurons probed with complex stimuli.

Chapter 3

Methods

3.1 Electrophysiology

Sprague-Dawley rats (postnatal day 17-25) were anesthetized with ketamine (60 mg/kg) and medetomidine (0.48 mg/kg) in strict accordance with the National Institutes of Health guidelines. After anesthesia, small craniotomy and durotomy were performed above the left auditory cortex. The brain cortex was covered with physiological buffer containing (in mM): 127 NaCl, 25 Na2CO3, 1.25 NaH2PO4, 2.5 KCl, 1 MgCl2, mixed with 1.5% agar. Temperature and depth of anesthesia were monitored throughout the experiment, and supplemental anesthesia was provided when required.

Standard blind whole-cell patch-clamp recording techniques modified from brain slice recordings (Stevens and Zador, 1998) were used to record subthreshold responses of single neurons to sounds. Membrane potential was sampled at 4 kHz, or 10 kHz in current clamp ($I = 0$) mode using an Axopatch 200B amplifier. Electrodes were pulled from filamented, thin-walled, borosillicate glass (outer diameter, 1.5 mm, inner diameter 1.17 mm) on a vertical two-stage puller. In some experiments, the internal solution, pH 7.25, diluted to 290 mOsm, contained QX-314; an intracellular sodium channel blocker for blocking action potentials. Resistance to bath was 3–5 MΩ before seal formation.

Recordings were made from primary auditory cortex (A1) as determined by anatomical landmarks and physiological criteria: tonotopic gradient and 'V-shaped' frequency-amplitude tuning properties of cells and local field potentials. Recordings (n=29 neurons) were made in superficial layers (approx. 200-600 μm, as determined

from micromanipulator travel. Detailed methodology can be found, for example, in Machens et al. (2004).

3.2 Stimuli

Natural stimuli taken from commercially available audio CDs were used as the main stimulus ensemble in this study. The stimuli were originally sampled at 44.1 kHz and resampled at 97.656 kHz for stimulus presentation. Animal vocalizations were taken from *The Diversity of Animal Sounds* and *Sounds of Neotropical Rainforest Mammals* (Cornell Laboratory of Ornithology, Ithaca, NY). The beginning sequence of *Purple Haze* (Jimi Hendrix) was taken from audio CD. The peak amplitude of each segment was normalized to the ± 10 V range of the speaker driver. A 20 ms cosine-squared ramp was applied at the onset and termination of some sound segments. The ensemble of natural sounds consisted of 122 sounds. All stimuli were presented free-field in a double-walled sound booth with the speaker facing the contralateral ear. The stimuli were delivered at 97.656 kHz using a System 3 RP2.1 with an electrostatic speaker (TDT, Alachua, Fl). The speaker had a maximum intensity (at 10 V command voltage) of 92 dB sound pressure level (SPL), and its frequency response was flat from 1 to 22 kHz to within an SD of 2.7 dB.

3.2.1 Stimulus representation

Upon reaching the cochlea in the inner ear all sounds are decomposed into their frequency components, and this information is transmitted further to the central nervous system. To approximate cochlear transformation of sounds into frequency components, all natural sounds were transformed into spectrograms (Fig. 3.1). Thus, the sounds were transformed into time-frequency domain using short-term Fourier transform (Klein et al., 2000). The spectrogram is given by the energy density spectrum of the sound pressure wave $s(t)$:

$$P(t,f) = \left| \frac{1}{2\pi} \int d\tau e^{-i2\pi f} s(\tau) h(\tau - t) \right|^2, \tag{3.1}$$

where t is time, f is frequency, and h is a window function. Here we used the Hamming window function (Press et al., 1992). For analysis we discretized the

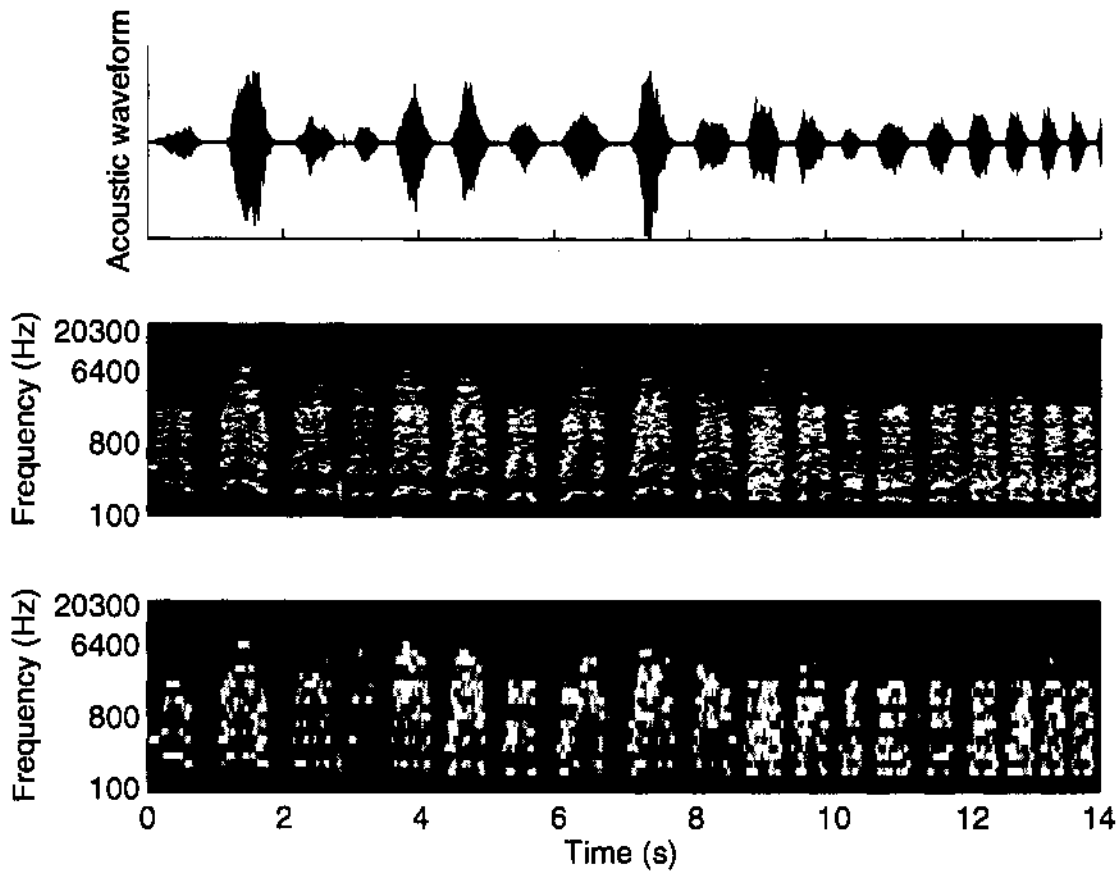

Figure 3.1: Acoustic waveforms of the stimuli were transformed into spectrograms, which served as input data set. Top panel shows the time varying amplitude of acoustic waveform (*Jaguar mating call*). Middle panel shows corresponding spectrographic representation of the waveform, with resolution of $\Delta x = 12$ frequencies per octave, $\Delta t = 5ms$; see text for details. Bottom panel shows spectrogram of the same sound computed with the actual resolution used in this study, *i.e.*$\Delta x = 3$, and $\Delta t = 10ms$.

spectrogram in both time and frequency. For time discretization we used a time window of $\Delta t = 10ms$. Frequency space was discretized on a logarithmic scale with $\Delta x = 3$ frequencies per octave (one octave is a doubling of frequency).

The discretized spectrogram $S(t, f)$ was then computed as:

$$S(t_i, f_l) = 20log_{10}\left[\int_{t_i}^{t_i+\Delta t} dt \int_{f_l}^{f_l^{(1+\Delta x)}} df\, P(t, f)\right], \tag{3.2}$$

where t_i with $i = 1 \ldots M$ are equally spaced time steps, and f_l with $l = 1 \ldots L$ are logarithmically spaced frequency steps.

The discretized spectrograms of all natural stimuli were then further processed to decrease computational demands and increase efficiency. Specifically, to decrease size of the input data set we used *principal component analysis* (PCA) to compute principal components of spectrograms.

Principal component analysis is a linear transformation that transforms given data to a new coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on. The low-order components thus contain most of the variance of the data, *i.e.*the "most important" aspects of the data. By keeping the lower-order principal components of natural sounds and ignoring higher-order ones, we reduced the dimensionality of the stimuli.

The principal components of spectrograms were computed as follows (Fig. 3.2). First, spectrograms were divided into 200 ms long overlapping segments. This arrangement provided us with a sequence of sound snippets, each containing 24 frequency bins and 20 time bins (given our choice of Δx and Δf above). Each segment was then considered a single data point in the input data set, with each such data point containing 24 times 20 'sound pixels.'

To reduce the dimensionality of the stimuli we then performed singular value decomposition of the set of all sound snippets, and projected the snippets onto the first 100 principal components. The first 100 principal components accounted for 98 % of the variance on average. Using more than 100 principal components dramatically increased network convergence time, but did not significantly change the final network solution.

The dynamics of natural stimuli usually introduce a low frequency bias (Field, 1987). To remove the bias (*i.e.*to 'whiten' the stimulus), the projections of sound snippets into principal component domain were scaled to have unit variance and zero mean.

3.3 Neural responses

Subthreshold sound-evoked responses were recorded using patch-clamp whole-cell recording technique. These subthreshold responses were then used as target data in the neural network model. The responses were resampled with the same time resolution as their corresponding input data sets (spectrograms). After re-sampling the response vector was scaled to have zero mean and unit variance.

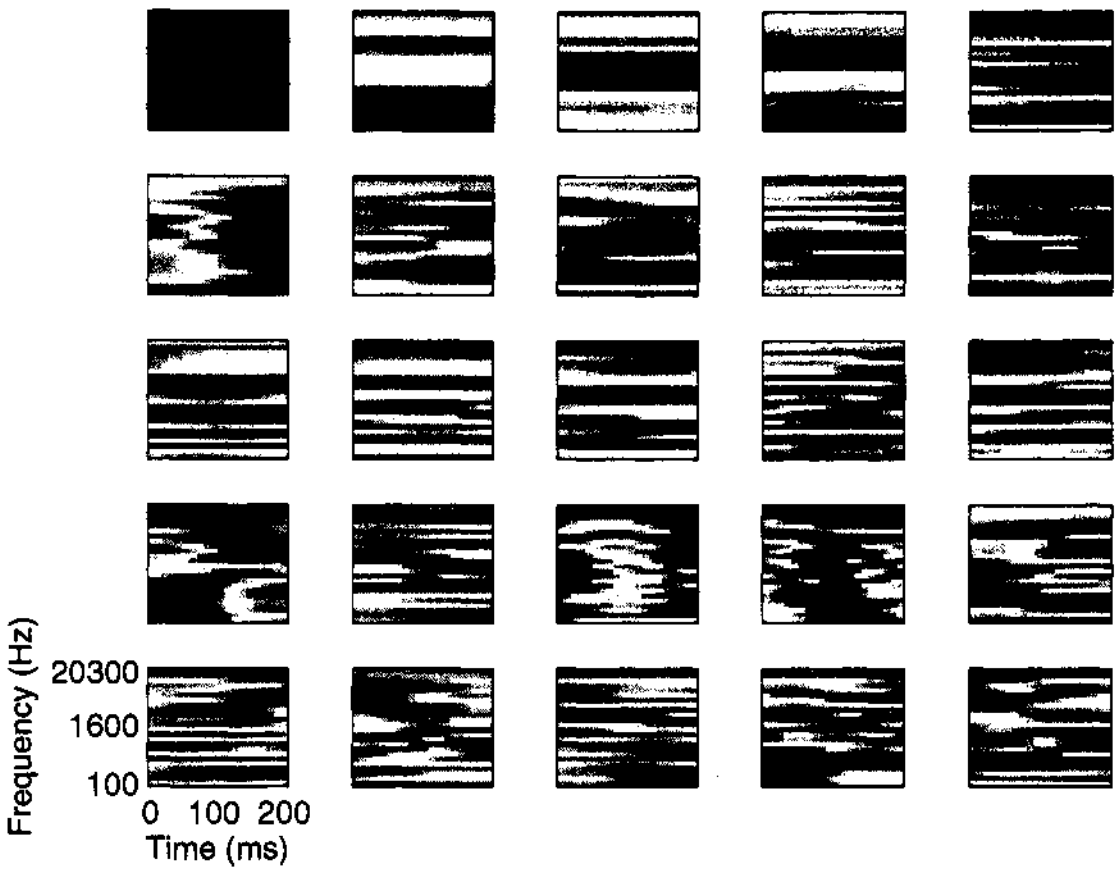

Figure 3.2: Principal components of the stimulus sequence were computed using principal component analysis. Figure shows the first 25 principal components of one stimulus sequence (the actual network input was computed using the first 100 principal components, which contributed 98 % of variance on average). Each component is depicted as 200 ms long spectrogram, where red corresponds to increase in component power, and blue to decrease in component power. Each 200 ms long portion of the original stimulus can be expressed as linear combination of the principal components.

3.4 Neural network

3.4.1 Network architecture

We used a two-layer feed-forward neural network with a tapped delay line architecture. The network consisted of input layer, hidden layer, and output layer (Fig. 3.3).

The input layer received a vector of normalized principal components of the natural sound stimuli (Sec. 3.2.1), and passed them onto the hidden units. Each hidden

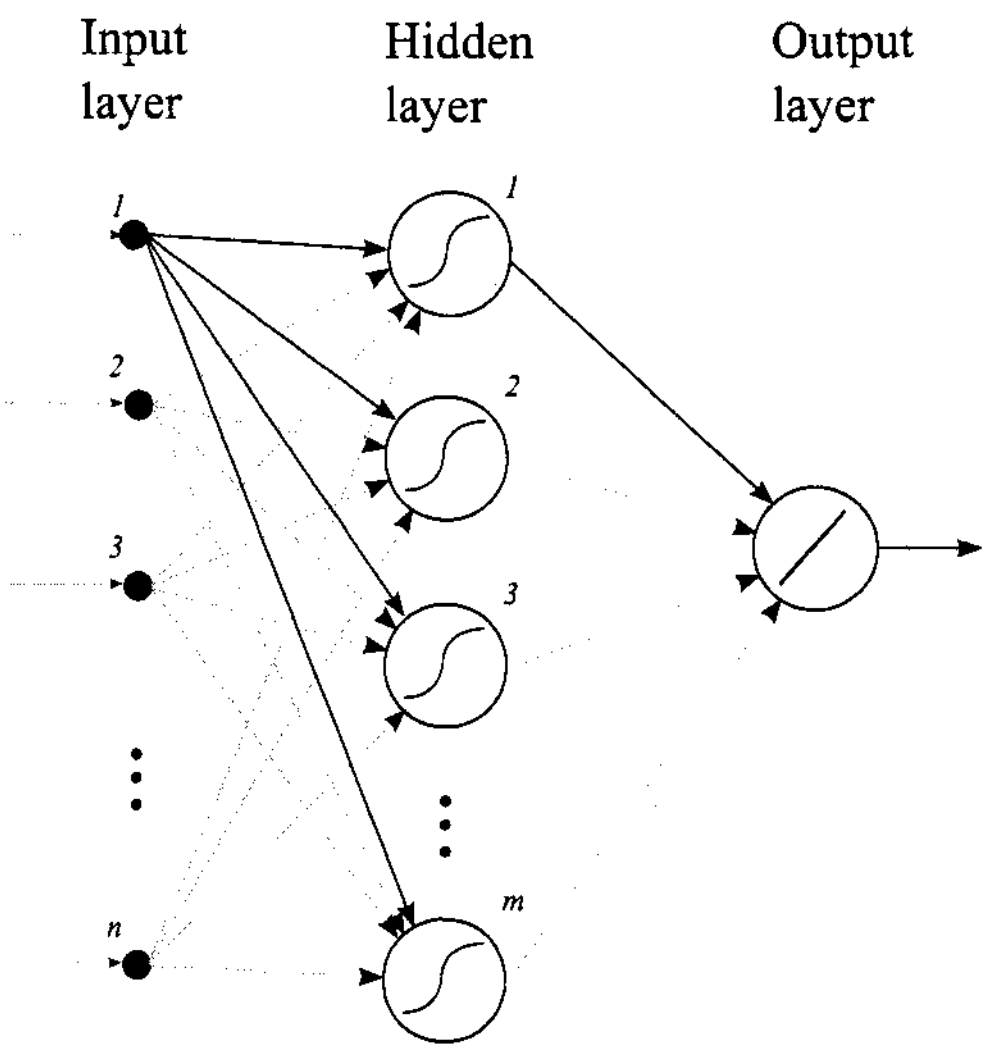

Figure 3.3: Network architecture. We used feed-forward network with one hidden layer and one output unit. We used $n = 100$ input units, corresponding to projections of spectrogram portions onto the first 100 principal components of the stimulus sequence. The network was initialized with $m = 12$ hidden units, which were pruned during training (see Sec. 3.4.6). The single output unit had a linear activation function.

unit then computed a weighted sum of its inputs, transformed the result with hyperbolic tangent as its activation function, and passed it onto the single output unit. The output unit computed a weighted sum of its inputs (from hidden units) which represented a prediction of subthreshold neuronal response to the given sound.

Thus the network computation can be written as (compare with Eqs. 2.3–2.4):

$$\hat{r}_{est}(t) = \sum_{j=0}^{m} w_j^o \tanh\left(\sum_{i=0}^{n} s_i w_{ji}^h\right), \tag{3.3}$$

where s is the input vector, $\hat{r}$ is the (predicted) response, w_{ji}^h are the input weights

(*i.e.*the weights associated with the hidden units), w_j^o are the output weights (associated with the output unit), n is the length of the input vector, and m is the number of hidden units. As usual, w_{j0}^h and w_0^o represent biases of their corresponding units.

For the analysis the input vector length was set to $n = 100$ first principal components of 200 ms long time window. With neural responses sampled at $\Delta t = 10ms$, each network output value corresponded to the mean of 10 ms long subthreshold response to the preceding 200 ms of sound. The network was initialized with $m = 12$ hidden units, some of which were removed during training (Sec. 3.4.6).

3.4.2 Weight initialization

Initialization of neural network parameters represents a very important step in neural network learning. Ad-hoc initialization could place the starting point of optimization away from the actual error surface minimum, and thus both prolong training and prevent the network from reaching a reasonable solution. On the other hand, efficient initialization can significantly speed up the convergence process.

Weights and biases in both layers were initialized using the Nguyen-Widrow weight initialization algorithm (Nguyen and Widrow, 1990). The algorithm calculates the interval from which the weights are taken in accordance with the length of the input vector and the number of hidden units. First, the weights are initialized with small random values. Then, the input space (input vector) is divided into small slightly overlapping intervals, the number of which is determined by number of hidden units. Finally, the weights are modified, such that each unit is assigned one of the input space intervals.

Thus the initial weight and bias values for each layer are set so that the active regions (*i.e.*intervals in which the activation function is not saturated) of the neurons are distributed approximately evenly over the layer's input space (Demuth and Beale, 2005). Therefore, more neurons are assumed to participate in optimization, because the active regions of all neurons are in the input space. During training, each unit can still adjust its weights and bias, but the weight adjustments are assumed to be small because the big adjustments were already eliminated during initialization.

3.4.3 Learning algorithm

We used *scaled conjugate gradient* algorithm (Møller, 1993) to optimize weights and biases of the network. The algorithm belongs to the class of conjugate gradient variants of the basic backpropagation algorithm (Rumelhart et al., 1986). The basic steepest descent algorithm searches along the negative of the gradient, which does not necessarily produces the fastest convergence. Conjugate gradient algorithms improve performance of the steepest descent algorithm by searching along conjugate directions. Such a search produces generally faster convergence than steepest descent directions.

The scaled conjugate gradient uses the curvature of the local search space to compute the optimal step size along the conjugate direction. The curvature is usually described by Hessian matrix, *i.e.*matrix of second-order derivatives of the performance function at the current values of network parameters (weights and biases). However, the evaluation of the Hessian matrix is plagued with computational difficulties (Haykin, 1999). Therefore, the scaled conjugate gradient algorithm takes advantage of the Levenberg-Marquardt approach to approximate the Hessian matrix: when the performance function has the form of a sum of squares, then the Hessian matrix can be approximated using the Jacobian matrix containing first derivatives of the network errors at the current values of weights and biases.

The basic step of a conjugate gradient algorithm can be written as:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \alpha_k \boldsymbol{p}_k, \tag{3.4}$$

where $\boldsymbol{w}_k$ stands for weight vector in the k^{th} iteration, α_k is the learning rate, and $\boldsymbol{p}_k$ stands for the gradients in the k^{th} iteration.

Computation of α_k requires computation of the Hessian matrix $\boldsymbol{H}(\boldsymbol{w}_k)$. In the scaled conjugate gradient algorithm this step is approximated as follows (Møller, 1993):

$$\boldsymbol{H}(\boldsymbol{w}_k)\boldsymbol{p}_k \approx \frac{\boldsymbol{J}(\boldsymbol{w}_k + \sigma_k \boldsymbol{p}_k) - \boldsymbol{J}(\boldsymbol{w}_k)}{\sigma_k}, 0 < \sigma_k \ll 1, \tag{3.5}$$

where $\boldsymbol{J}(.)$ stands for the Jacobian matrix. The Jacobian matrix can be computed through a standard backpropagation technique that is much less complex than computing the Hessian matrix. Doing so the scaled conjugate gradient algorithm combines the advantages of conjugate gradient search with computational efficiency.

3.4.4 Regularization

One of the common problems during neural network training is *overfitting*. Network is said to overfit the training data set when the error on the *training* data set is small, but the error on a *validation* data set (*i.e.*new data not used for training) is large. Then the network has virtually memorized the training data set but has failed to generalize to new data. And although an initial overfit might be useful to 'guide' the network on the error surface (Sec. 3.4.6), in general overfitting is undesirable and must be prevented.

To prevent overfitting we used *regularization* with a modified performance function. The usual performance function used in feed-forward neural networks is the mean squared error (see also Eq. 2.5). We have used a modified performance function which included the mean of the sum of squares of the network parameters (weights and biases):

$$E_{reg} = \frac{\gamma}{n}\sum_{i=1}^{n}(\hat{r}_i - r_i)^2 + \frac{(1-\gamma)}{N}\sum_{j=1}^{N} w_j^2, \tag{3.6}$$

where $\hat{r}$ is the predicted response, r is the actual response, n is the length of the input vector, $N = nm + 2m + 1$ (see Eq. 3.3) is the total number of network parameters, w_j is a network parameter (weight, or bias), and γ is the *performance ratio*.

The second term in Eq. 3.6 described contribution of weights and biases toward the network performance, and the performance ratio described the weight of this contribution. We have used $\gamma = 0.5$ to force equal contribution of errors and network parameters toward the network performance. This performance function caused the network to have smaller weights and biases (as opposed to using the traditional mean squared error), with the network response smoother and less likely to overfit.

3.4.5 Network training

The initial network with 12 hidden units was trained using the scaled conjugate gradient algorithm (Sec. 3.4.3). The training was stopped either after 500 training epochs or when the network error fell below 0.2, whichever occurred first. We

assumed that procedure enabled a reasonable overfit of the training data. Henceforth when we say that the *training has converged* we mean that the network has either been trained for 500 epochs, or reached error less than or equal to 0.2.

After this initial training one of the hidden units was pruned (Sec. 3.4.6) and training continued with a new network with fewer hidden units until the training converged. This procedure has been repeated until we were left with a network with only one hidden unit. The best network out of the resulting 12 networks was then selected as the trained network (Sec. 3.4.6).

To avoid the learning algorithm being trapped in local minima, we repeated the training procedure ten times, each time using different initial values of network parameters. The ten training cycles then left us with ten pruned networks, and the network with minimum squared error on reserved training data set (pruning data set) was declared a winner.

3.4.6 Hidden unit pruning

The neural network was initialized with 12 hidden units. This number was chosen to provide the network with a sufficient number of parameters to obtain overfit the data initially, *i.e.*to obtain a very good fit to the *training data*. However, networks which overfit training data generalize poorly. Also, when the input-output transformation approximated by the network is described by too many parameters, these parameters are hard to interpret. We used pruning procedure to find a minimum number of hidden units (for the given network) that approximated the data well (Prenger et al., 2004).

After the initial training converged for network with 12 hidden units, one of the hidden units was removed from the network. Then the *output* weights and output bias of the new network were optimized. Because the output unit used a linear activation function, it formed a linear filter, and we used the least mean squares algorithm, or Widrow-Hoff algorithm, to minimize the mean squared error. This optimization procedure was repeated 12 times, each time removing a different hidden unit. The network with the best performance (with 11 hidden units) was then selected and training continued (with the *original* hidden network weights and *updated* output unit weights) until the network converged. The pruning process was then repeated until the final network with only one hidden unit.

Performance of the resulting 12 networks (with 1–12 hidden units) was then evaluated on 10 % of the input data set, which was not using for training (pruning data

set). Network with the smallest mean squared error on this reserved data set was then selected as the final trained network which best described the input-output transformation of the neuron.

3.4.7 Evaluating predictions

The true ability of the network to generalize and learn the input-output transformation of a neuron can only be tested with a new data. As mentioned above, achieving an extremely small error on training data set can lead to overfitting and consequently poor generalization. Hence, to evaluate the true performance of our network we selected 10 % of the data set as a *validation data set*, which was never used in either training or pruning. The predictive power of the network was then determined by computing the correlation coefficient between observed neuronal responses and network predictions.

3.4.8 Network interpretation

A trained neural network represents represents a solution to a nonlinear regression problem, *i.e.*transformation of a sensory stimulus into a neuronal response for a given neuron. We have used visualization of the weights of the hidden units to interpret this nonlinear transformation of each neuron (Lau et al., 2002; Lehky and Sejnowski, 1990; Lehky et al., 1992; Prenger et al., 2004). For networks with many hidden units, visualization and consequent interpretation would be cumbersome. We have therefore implemented pruning algorithm (Sec. 3.4.6) to decrease the number of hidden units and facilitate their interpretation.

Each hidden unit acted as a filter of the input vector, with the filter properties specified by unit's set of weights. We wished to interpret such a filter as the primary feature of the input space transformed by the neuron. Recall that each hidden unit received input from an input vector consisting (at each time step) of 100 normalized pca components of a 200 ms long sound snippet (Sec. 3.2.1). To visualize the hidden weights, we first 'denormalized' set of weights for each hidden unit using the original mean and variance of the first 100 pca projections of the sound stimuli. The weights were then considered equivalent to pca projections and projected back to the 'spectrogram' space of the original sound stimulus. The result of this transformation was then interpreted as the primary acoustic feature of the stimulus set recognized by the hidden unit(s).

Although visualization of weights does not provide us with all information about network function, for example interactions between hidden units, it nevertheless proved to be very valuable for interpretation of stimulus features represented by the neuron, and relationship of the nonlinear regression model (provided by the network) to a linear model for the same neuron.

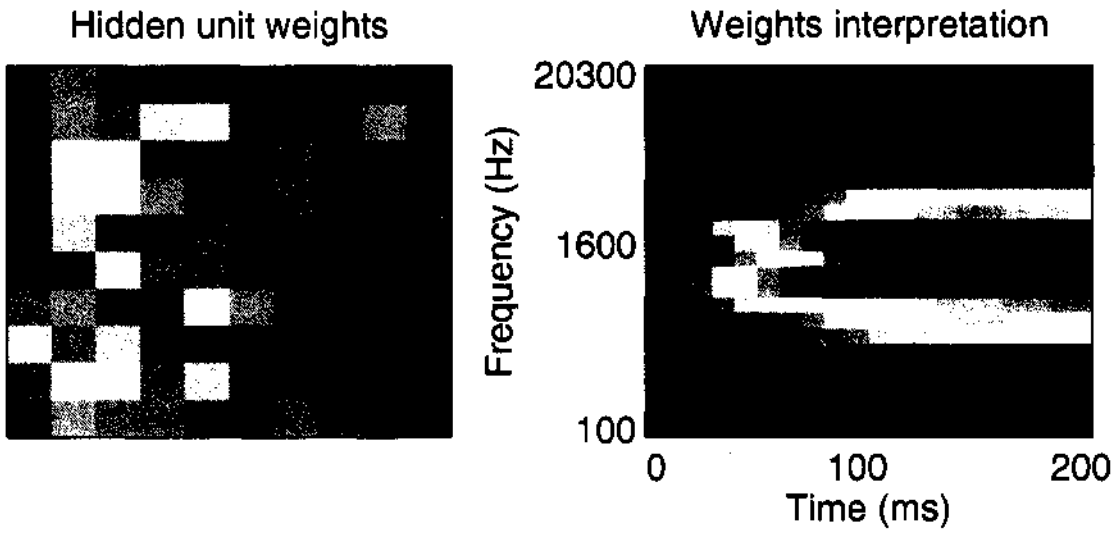

Figure 3.4: Weights associated with each hidden unit (left panel) were projected back to spectrograms (right panel). Left panel shows set of weights associated with one randomly chosen hidden unit. For purpose of this figure, 100 weights were reordered column-wise. These weights were then projected from the space defined by the first 100 pca components back to sound stimuli space. Right panel shows the projection of the left panel. This hidden unit was strongly activated by sounds containing frequencies in a relatively narrow band around 1600 Hz. Values in both panels were independently normalized, so that blue corresponds to -1 and red corresponds to 1.

3.5 STRF estimation

The spectro-temporal receptive field (STRF, Sec. 2.3.3) is a linear description of the behavior of a neuron (deCharms et al., 1998; Kowalski et al., 1996, and many others). We estimated STRFs for the neurons in our sample to provide us with estimates of linear components of neuronal transformation. STRFs were estimated using multidimensional liner regression, for details of this procedure see (Machens et al., 2004).

Using STRF, neuronal response was estimated by linear filtering of the stimulus spectrogram $S(t_i, f_l)$ (Eq. 3.2) with the STRF $H(-t_k, f_l)$ of the neuron:

$$\hat{r}(t_i) = \sum_{k=1}^{K} \sum_{l=1}^{L} H(-t_k, f_l) S(t_i - t_k, f_l), \tag{3.7}$$

where $\hat{r}$ is the estimated response; $-t_k$ uses a negative time index to comply with the conventions of the reverse correlation approach. STRF has a finite temporal extent, in our case $K = 20$ covering 200 ms of sound (given $\Delta t = 10ms$ temporal resolution of the spectrogram). To use linear regression we simplified the notation as follows: We used $\hat{r}_i = \hat{r}(t_i)$, re-ordered indices such that $a_j = H(-t_k, f_l)$ and $s_{ji} = S(t_i - t_k, f_l)$ with $j = (l-1)K + k$. Both response and stimuli were normalized to have zero mean. Eq. 3.7 the simplified to:

$$\hat{r}_i = \sum_{j=1}^{N} a_j s_{ji}, \tag{3.8}$$

where $N = KL$. The STRF was now given as a_j which was fitted by minimizing the mean square error between the estimated response $\hat{r}_i$, and the measured response r_i. The solution is provided by multidimensional linear regression:

$$a_j = \sum_{k=1}^{N} B_{jk}^{-1} A_k, \tag{3.9}$$

where $A_k = \frac{1}{M} \sum_{i=1}^{M} s_{ki} r_i$ is the stimulus-response cross-covariance, and $B_{jk} = \frac{1}{M} \sum_{i=1}^{M} s_{ji} s_{ki}$ is the stimulus-stimulus autocovariance.

We used ridge regression (Hastie et al., 2001) as a regularization approach to penalize strong deviations of the parameters from zero. With ridge regression Eq. 3.9 transformed into:

$$a_j = \sum_{k=1}^{N} (B + C)_{jk}^{-1} A_k, \tag{3.10}$$

where $C_{jk} = \lambda \delta_{jk}$, with λ denoting the strength of the constraint, and δ_{jk} denoting the Kronecker δ with $\delta_{lk} = 1$, if $l = k$ and $\delta_{lk} = 0$, otherwise.

The training data set used to train the neural network was used to fit the STRF, and STRF predictions were evaluated on the validation data set used for evaluating network response predictions. The STRF as a linear model should then perform similarly to a neural network with one hidden unit with a linear activation function.

Chapter 4

Results

The primary goal of this thesis was to use artificial neural networks to study stimulus-response function of single auditory neurons presented with a spectrally rich ensemble of natural sounds. We sought to describe the nonlinear mapping function in terms of the 'preferred' acoustic features (acoustic kernels) which would be able to describe the transformation from sound to neuronal response. We were particularly interested in the ability of neural network model to predict neuronal responses, and we compared these predictions with predictions made by a linear model.

Our analysis consisted of several steps. First, the sound stimuli and recorded data were preprocessed to remove excessive variability. Second, we trained neural networks in order to approximate the stimulus-response mapping function. Third, we computed the linear component of this mapping function (STRF) for each neuron. Finally, we compared response predictions made by the (nonlinear) neural network model and the linear model.

4.1 Data characterization

We used data recorded from 29 neurons in the primary auditory cortex using in-vivo patch-clamp whole-cell recordings. Action potentials were either blocked using the intracellular sodium channel blocker QX-314 (n=13 cells), or filtered out using a median filter (n=16 cells). The used neural responses then consisted only of fluctuations in the subthreshold membrane potential, which reflected the total

synaptic input to the neuron. Thus any nonlinearities in the stimulus-response relationship, *i.e.*any deviations from the linear model, could not be attributed to the all-or-nothing transformation of membrane potential to action potential (spike). For analysis we resampled the responses with time resolution $\Delta t = 10ms$, corresponding to the time resolution of the stimuli (see below and Sec. 3.2.1).

Sound stimuli consisted of various animal communication calls and environmental sounds that lasted for 8–15 seconds. The stimuli were selected to be representative of the acoustic environment of rats which were used for this study. We used two separate sets of natural sounds. The first set consisted of different combinations of several natural sound blocks presented in pseudorandom order. The second set consisted of a wider range (n=122) of sound segments, of which a subset was tested on any particular neuron. For detailed description of responsiveness of neurons, as well as description of a subset of the stimuli see (Machens et al., 2004).

For analysis, we transformed the temporal waveforms of the sound stimuli to spectrograms (Fig. 3.1, see also Sec. 3.2.1). This transformation provides a rough approximation of the first stage of auditory processing in the cochlea, when the sound pressure wave is transformed into frequency components as functions of time. The example spectrogram in Fig. 3.1 demonstrates the basic features of natural sounds that distinguish them from artificial stimuli. Namely, natural sounds contain long range correlations in frequency and time, and are nonstationary, *i.e.*their statistical properties fluctuate over time, see also (Machens et al., 2004).

We then binned the spectrograms into 200 ms long overlapping windows, sliding by 10 ms 'across' spectrogram. Each window then served as one input vector to the neural network, and the neuronal response recorded during the last 10 ms of any given window served as the associated output data point. This organization can be interpreted as a continuous sequence of (gradually changing) sound snippets (200 ms long) each evoking neuronal response at its end. According to this arrangement each input vector would contain 480 elements (24 frequency bins times 20 time bins). To reduce dimensionality of the stimulus, and consequently improve network training and convergence time we performed principal component analysis (PCA, Sec. 3.2.1) on the input data set. For each neuron (and each input data set) we identified the first 100 PCA components, which covered 98 % of variance on average. Fig. 3.2 show first 25 PCA components of one of the input data sets. It is evident from the figure that these components changed slowly in frequency and time. Since these components represent most of the variance in stimuli, using a subset of the most important components effectively low-passes the stimuli. Projections of the input data vectors on the first 100 PCA components then served as input data vectors. Increasing the number of components beyond

100 had little effect on the final network solution, but it increased network training time.

4.2 Network training

Our network model consisted of one hidden layer of units connected to one output unit, representing neuronal response. Each hidden unit received input from the input data vector (100 elements long, see above). Before training, each data set (input and output data) was divided into 3 continuous non-overlapping sets. The training data set consisted of 80 % of the data, pruning, and validation data sets each consisted of 10 % of the data. For each neuron, the neural network was initialized with 12 hidden units and trained on the full training data set (Fig. 4.1).

After the training converged, one hidden unit was removed from the network. We identified the hidden unit first by optimizing the output layer (in mean square error sense), each time with one hidden unit removed. The unit, removal of which led to the minimum error on the output, was then pruned from the network, because such a unit would increase the error in the consequent training.

After hidden unit pruning, the resulting network contained one less hidden unit, and was again trained on the full data set until the training converged. The whole process was repeated until we were left with network containing only one hidden unit. The training process thus consisted of training 12 networks with decreasing number of hidden units. After training 12 networks, the network performance was evaluated on the pruning data set, when responses of the 12 networks (with 1–12 hidden units) were compared to the actual pruning data set responses, and the network with the minimum squared error was identified as the final network.

The training procedure was repeated ten times, each time with different initial values of network weights for the network with 12 hidden units. Out of the ten neural networks selected from ten training cycles (note that each such network could have had different number of hidden units), we identified neural network with the best performance on pruning data set as the final neural network (Fig. 4.2).

4.3 Network predictions

We first trained networks on a subset of cells (n=16) using different combinations of blocks of natural sounds. Each network was trained on 90 % of the available

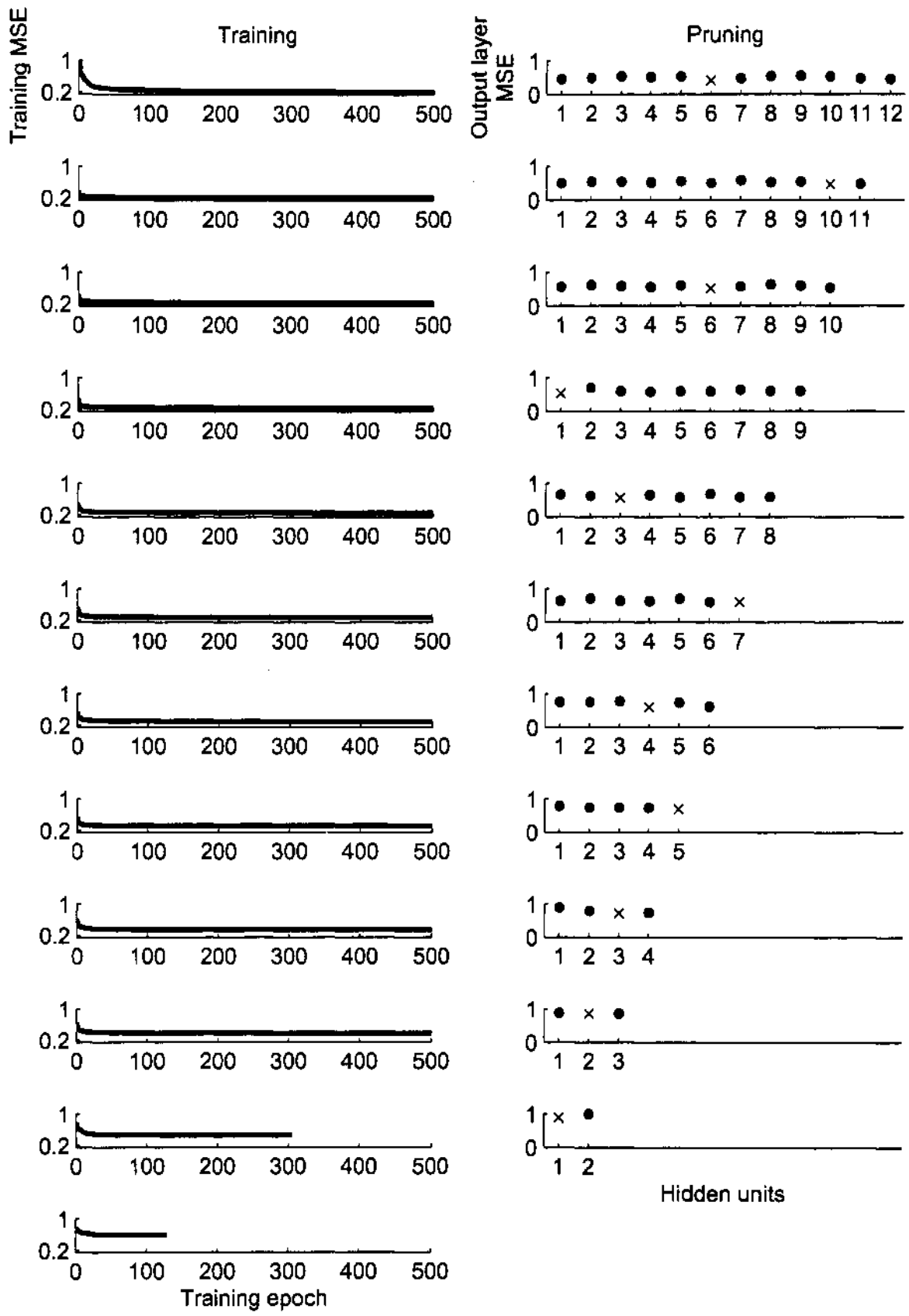

Figure 4.1: Each network was initialized with 12 hidden units, and trained 12 times with decreasing number of hidden units. Left panels show decrease in mean square error during training for networks with 12 (top) to 1 hidden unit (bottom). Right panels: after training for a given number of hidden units converged, one hidden unit was removed from the network. Graphs in the right panels show mean square error of the output layer with the corresponding hidden unit removed. Crosses mark hidden units which were pruned from the network.

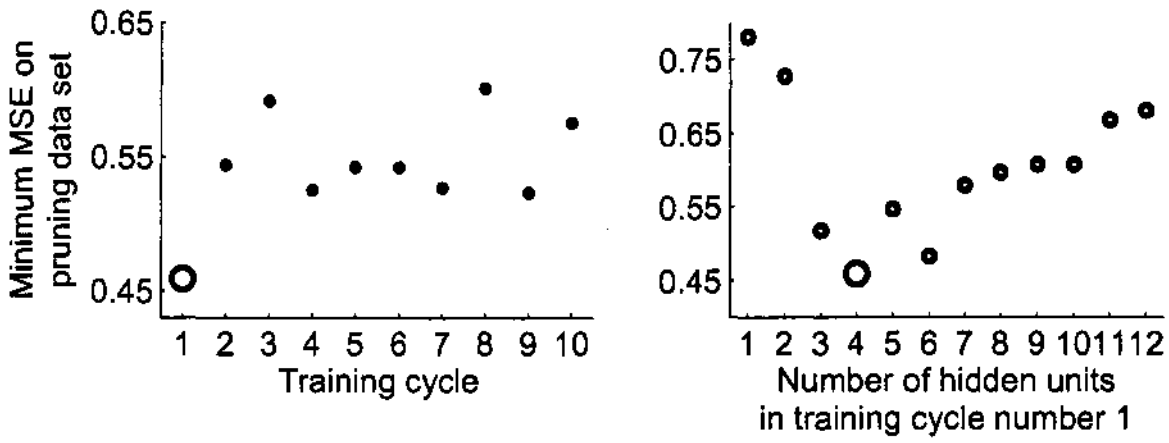

Figure 4.2: The final network was identified based on its performance on pruning data set. Left panel shows mean square errors of ten networks selected in ten training cycles. The large empty dot identifies the network with the best performance. Note that each of the ten networks can contain different number of hidden units. Right panel shows all mean square errors for 12 networks from the first training cycle. The large empty dot shows network with the best performance. Thus for this neuron, the final selected network was network with 4 hidden units from the first training cycle.

data (80 % training data and 10 % pruning data) An example neural network trained for one of these neurons is summarized in Fig. 4.3.

We first visualized the weights of the hidden units by projecting them back to the space of the natural sounds used as stimuli (see Sec. 3.4.8). We interpreted the projected weights as important acoustic features able to evoke and shape the neuronal response. As is evident from Fig. 4.3 the five acoustic features for this particular cell have their power concentrated around 3 kHz and mostly cover the whole 200 ms of our sound window.

Such an interpretation of weights is a simple way to visualize network parameters simultaneously and interpret them in the terms of input (sound) data space. It is, however, the non-trivial combination of these weights that determines the network output (*i.e.*neuronal response). The final contribution of each hidden unit is given by parameters of its activation function and its associated output weight. Activation functions of the network are plotted in Fig. 4.3 alongside their associated hidden units. The slope of each activation function was given by the gain (sum of weights) of each hidden unit (see also Eq. 3.3). The x- and y- positions of the activation functions were determined by input and output bias terms respectively, and the function amplitudes were given by the output weights. It is clear that this network described a complex relationship among the hidden units.

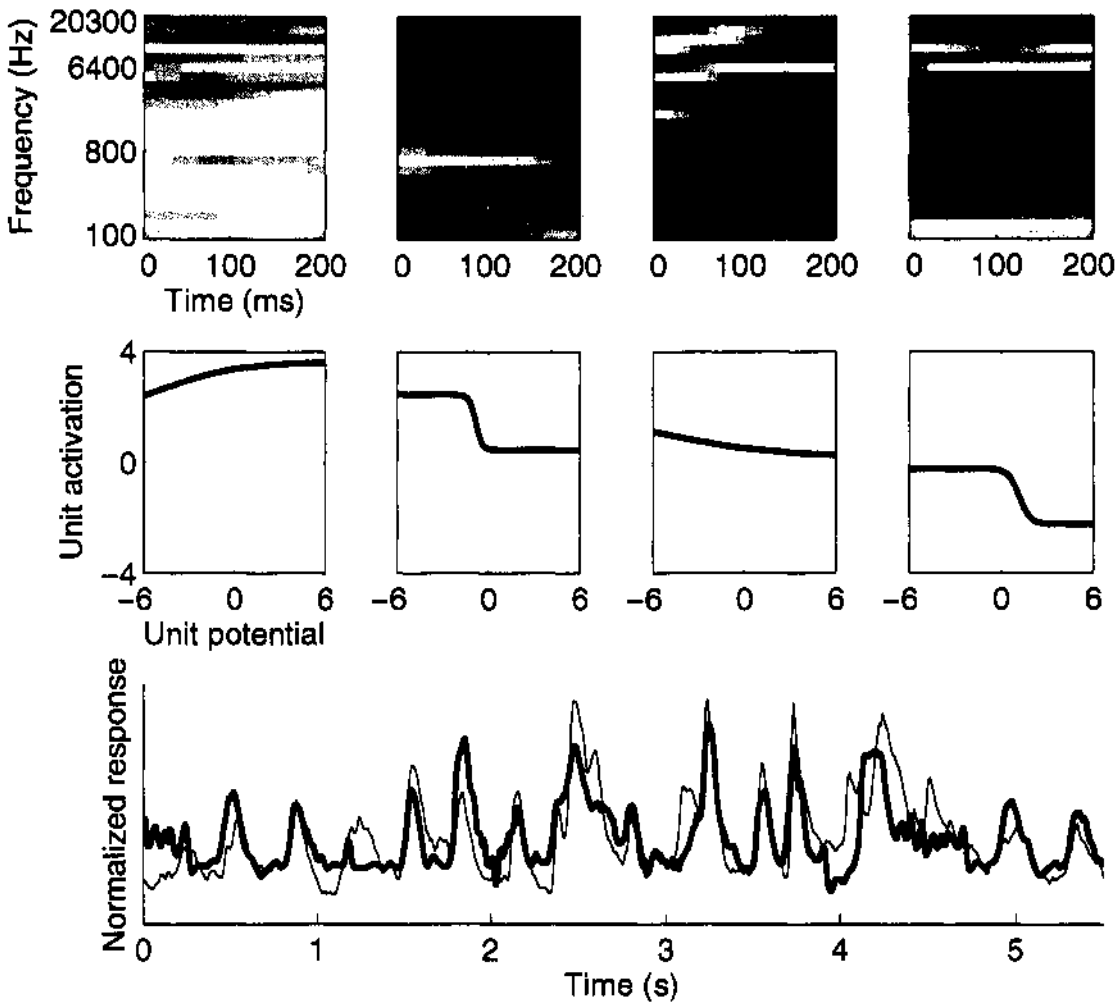

Figure 4.3: Neural network summary for an example neuron. Top panels show weights of hidden units interpreted as sound snippets, each of which is accompanied by its associated activation function (middle panels; see text for details). Bottom panel shows the actual neuronal response from the validation data set (thin line), and the response predicted by the network (thick black line).

The ultimate test of the neural network performance, however, is its ability to generalize, *i.e.*to predict responses to an unknown set of stimuli. We therefore tested the network on validation data set (a separate data consisting of 10 % of the total data set), which was used neither for network training, nor for network pruning. The predicted neuronal response to the validation data set is shown in Fig. 4.3.

The correlation coefficient between the neuronal response and predicted response was 0.65, confirming that the network captured the main features of the stimulus-response relationship. The figure also confirms that, in addition to capturing overall trend in the response, the network also captured finer details of the response. The correlation was far from perfect, however, which could have been determined

by several factors, such as noisy data, or simply inability of the training procedure to recover the perfect transformation. We will return to these issues in the discussion. Overall, the mean correlation coefficient between he actual and predicted response was 0.63 suggesting very good generalization capability of our network models.

We were wondering to what extent this good generalization could have been determined to our undersampling of the stimulus space. The neurons summarized in the previous paragraphs were probed with stimuli consisting of different blocks of natural sounds organized into pseudo-random sequences. Thus similar stimuli might have entered into training and validation data sets for the same neuron. This would have made the training and validation sets similar and the good performance might—to some extent—be due to overfitting on the training set.

We have therefore trained neural networks on an additional set of neurons (n=13) probed with even richer set of natural sounds. Summary of one example neural network is shown in Fig. 4.4.

In this case, the final network contained six hidden units and the correlation coefficient between the actual and predicted response was 0.63. The network was able to capture the overall trend in the data, as well as some of the finer details of neuronal response. Overall, the mean correlation coefficient for this data set was 0.34. Interestingly, the average number of hidden units for neural networks trained on this data set was 5.8, which was lower than the average number of hidden units for the previous data set (n hidden units=6.9), although not significantly lower. Regardless, this suggested that increasing the complexity of the stimuli might have indeed resulted in better generalization of the networks, reflected in lower number of important acoustic features recovered by the networks. This better generalization however was also accompanied by overall slight decrease in performance.

4.4 Comparison with the linear model

We have shown that neural networks can capture the general characteristics of stimulus-response functions of single neurons. However, we were also interested whether the neural network model captured some of the assumed nonlinearities in neuronal transformations. We have therefore compared performance of our neural networks to performance of linear models represented by the standard spectro-temporal receptive field (STRF) model (Sec. 3.5).

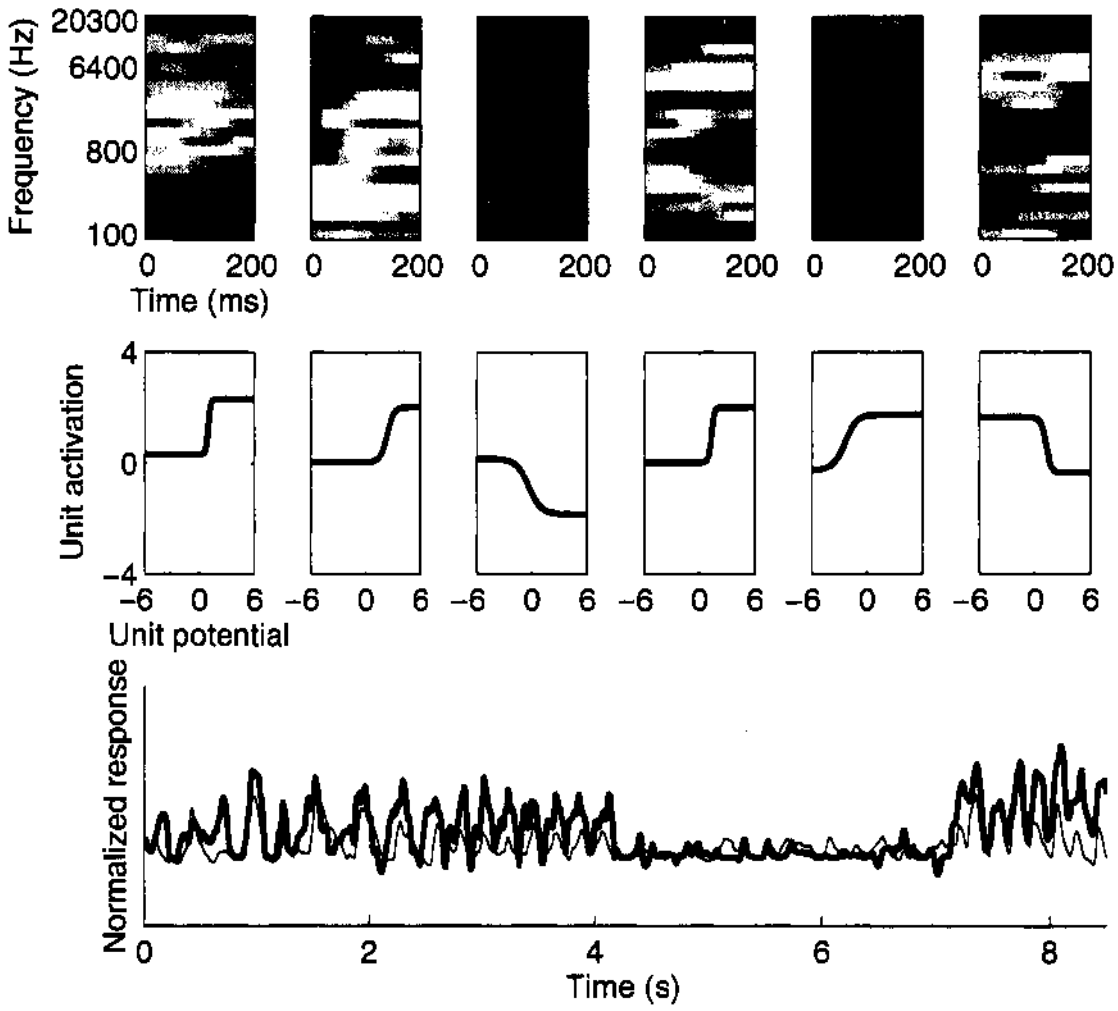

Figure 4.4: Neural network summary for an example neuron from a different data set. Top panel shows weights of hidden units interpreted as sound snippets, each of which is accompanied by its associated activation function (middle panel; see text for details). Bottom panel shows the actual neuronal response from the validation data set (thin line), and the response predicted by the network (thick black line).

STRF model captures the linear component of the stimulus-response relationship, and as such it has been usually estimated using the reverse-correlation method (for example in deCharms et al., 1998) on the basis of the well defined random stimuli. The natural stimuli we have used, however, featured correlations in both temporal and spectral domains. The reverse correlation approach was therefore generalized using linear regression by dividing the reverse-correlation solution by the autocovariance of the stimulus (see Eq. 3.9). We computed STRFs from the training and pruning data sets, and then computed predictions of neuronal responses from the validation data set. Both neural networks and linear models used the same data sets for training, and the same data sets for validation of performance. Performances of these two different classes of models were thus directly comparable.

Representative STRFs are shown in Fig. 4.5. The recovered STRFs typically featured an arrangement of excitatory (red) and inhibitory (blue) regions indicating times and frequencies at which stimulus energy led to increase or decrease in the neuronal response, respectively.

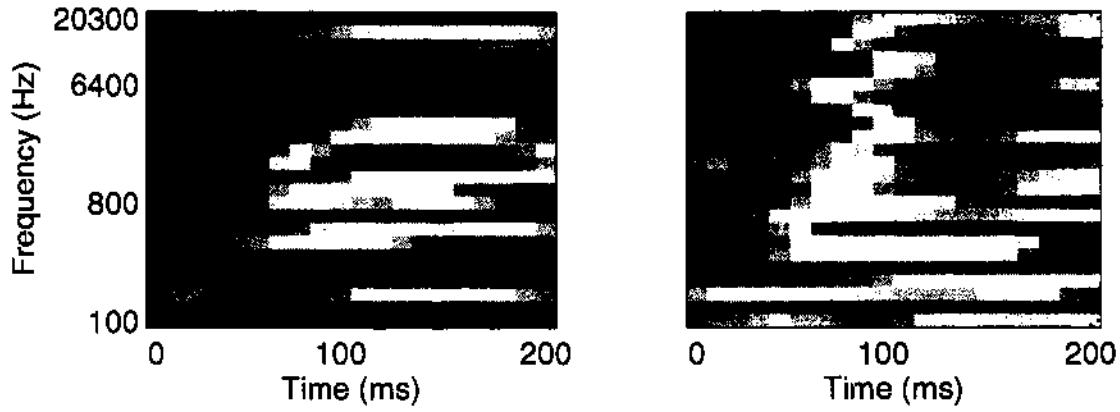

Figure 4.5: Linear components of stimulus-response relationships were captured by spectro-temporal receptive fields (STRF). STRFs visualize and quantify the linear relationship between sounds of particular frequencies at particular times, and the increase or decrease in response. Two panels show two examples of STRFs recovered from our data set. Left panel shows STRF with a narrow excitatory (red) band, *i.e.*for this neuron sounds containing frequencies around 1000 Hz tended to increase neuronal response. Right panel shows STRF with wide frequency tuning (spanning 600–20000 Hz). In our visualization (as spectrograms), each STRF can also be interpreted as the *linear* "preferred acoustic feature," which tends to increase neuronal response.

The performance of STRF models, however, was generally worse than performance of neural network models. Predictions of neuronal responses estimated with the linear model typically captured only the crude characteristics of responses (Fig. 4.6), and failed to capture the finer details and also the amplitude of response modulations.

On average, neuronal networks were able to capture finer details of neuronal responses than the corresponding linear models (Fig. 4.7). This ability of neural networks was determined by their multiple hidden units, which essentially corresponded to several linear models in parallel.

To compare performance of the non-linear neural network models and linear models we computed correlation coefficients between predicted and actual neuronal responses (Fig. 4.8, see also Sec. 3.4.7).

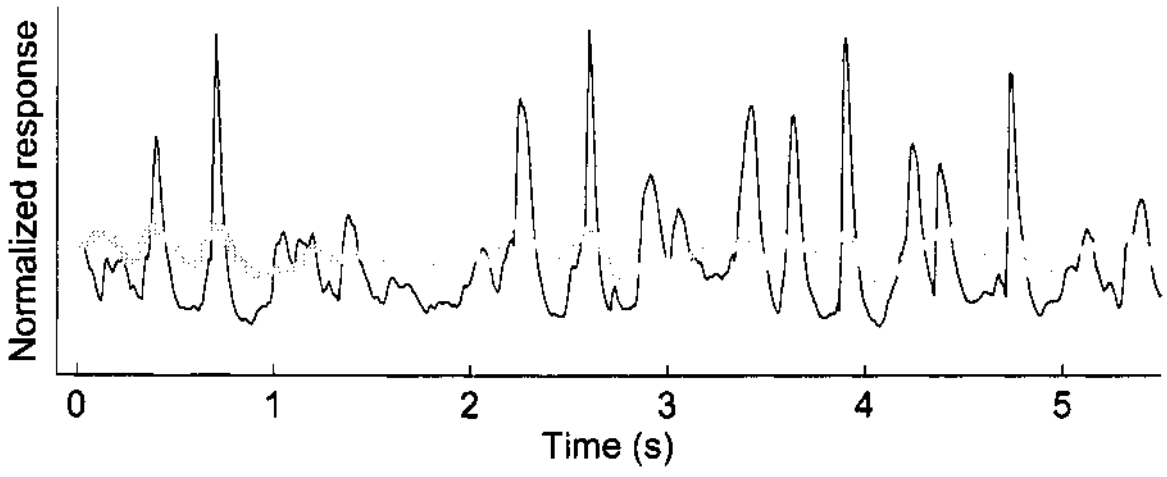

Figure 4.6: Neuronal responses predicted by the linear model (STRF) failed to capture finer details and amplitude of response modulations. For an example neuron, the actual (thin line) and predicted (thick gray line) responses are overlayed for comparison.

Fig. 4.8 summarizes performance of both neural network models and linear models. Neuronal networks showed significantly better performance in our neuronal population than linear models (two-tailed paired t-test, $p \ll 0.001$). Mean correlation coefficient between predicted and actual neuronal responses for neuronal networks was 0.50, and for linear models 0.31.

The multidimensional linear regression used to compute the STRFs is a special case of non-linear regression essentially performed by our neuronal networks during the training process. We were therefore interested to see whether the neural networks were able to recover 'linear' kernels in a form similar to STRFs. From the training cycle in which we selected the final network for each neuron, we identified the network with only one hidden unit and compared its input weights to STRF of the corresponding neuron. Two examples are shown in Fig. 4.9. Neural networks trained on natural sound stimuli and associated neuronal responses were also able to recover the linear component of neuronal response.

4.5 Results summary

We conclude that neural networks trained on natural sound stimuli provided a rich description of the nonlinear stimulus-response transformations of single neurons. The networks captured well both fine details and amplitude of neuronal responses (Fig. 4.3, Fig. 4.4), and the predictions were qualitatively better than predictions

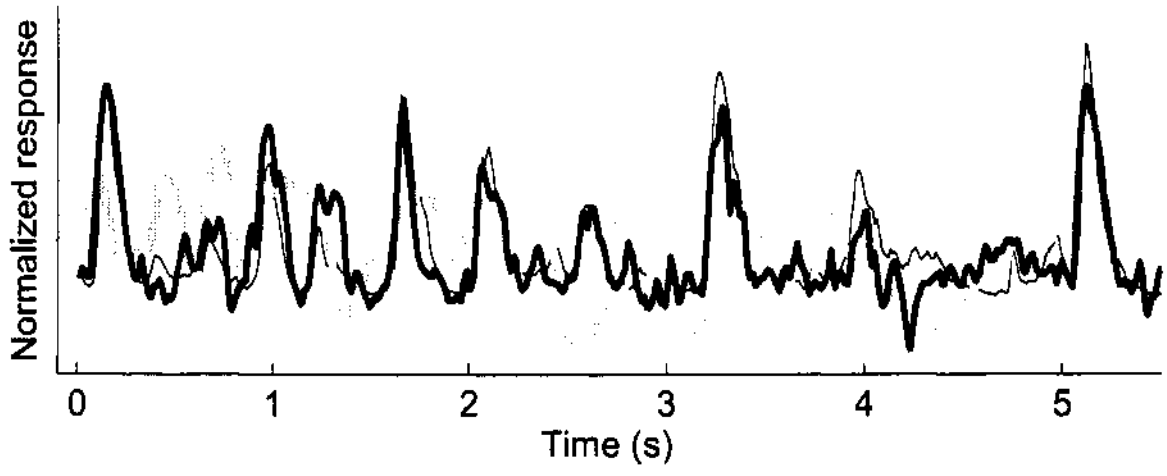

Figure 4.7: Neuronal networks were able to capture finer details of neuronal responses than STRF models. All responses (actual and predicted) were renormalized to have zero mean and unit variance, because STRF predictions did not capture the amplitude of neuronal responses. Thin line shows the actual neuronal response, thick black line the prediction made by neuronal network, and thick gray line prediction made by linear model (STRF).

made by linear models (Fig. 4.7). Neural networks outperformed the linear models also in quantitative terms with average prediction correlation coefficient of 0.50, significantly better than average correlation coefficient for predictions made by linear models (0.31, Fig. 4.8). The improvement in performance could be (partially) explained by neural networks enhancing the linear models by performing nonlinear regression. Indeed, the linear kernels of neurons were often recovered by the neural networks as well (Fig. 4.9).

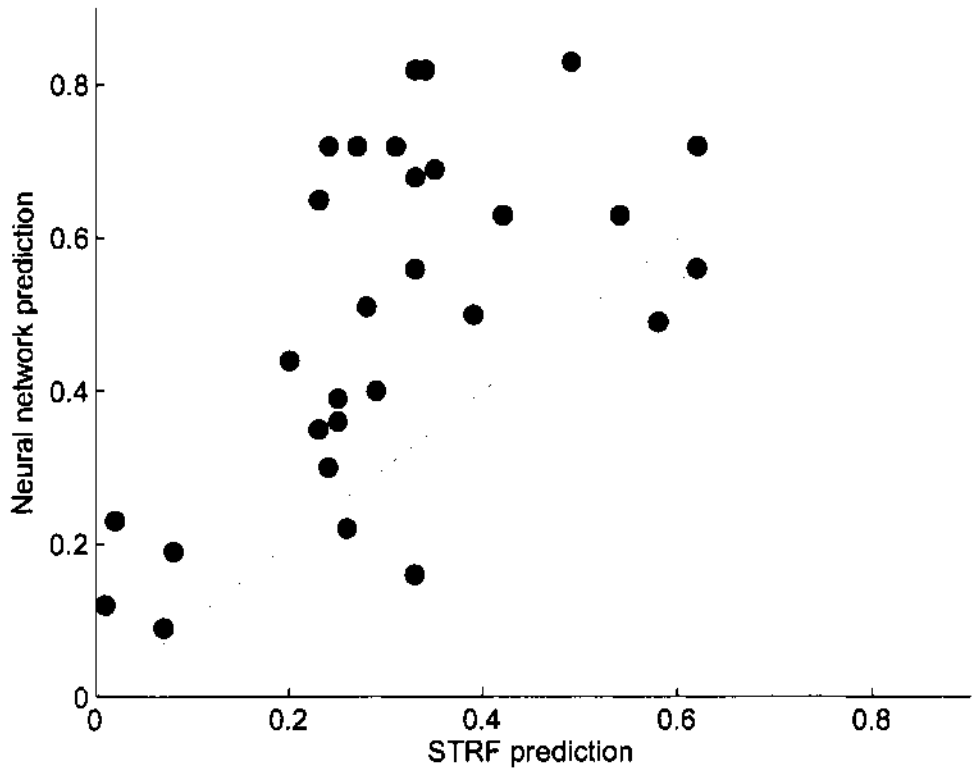

Figure 4.8: Neural networks outperformed linear models (STRFs) when predicting responses on validation data sets. Shown is comparison of correlation coefficients (n=29 neurons) between actual and predicted responses to validation data sets using either linear model (STRF prediction) or neural network model (Neural network prediction).

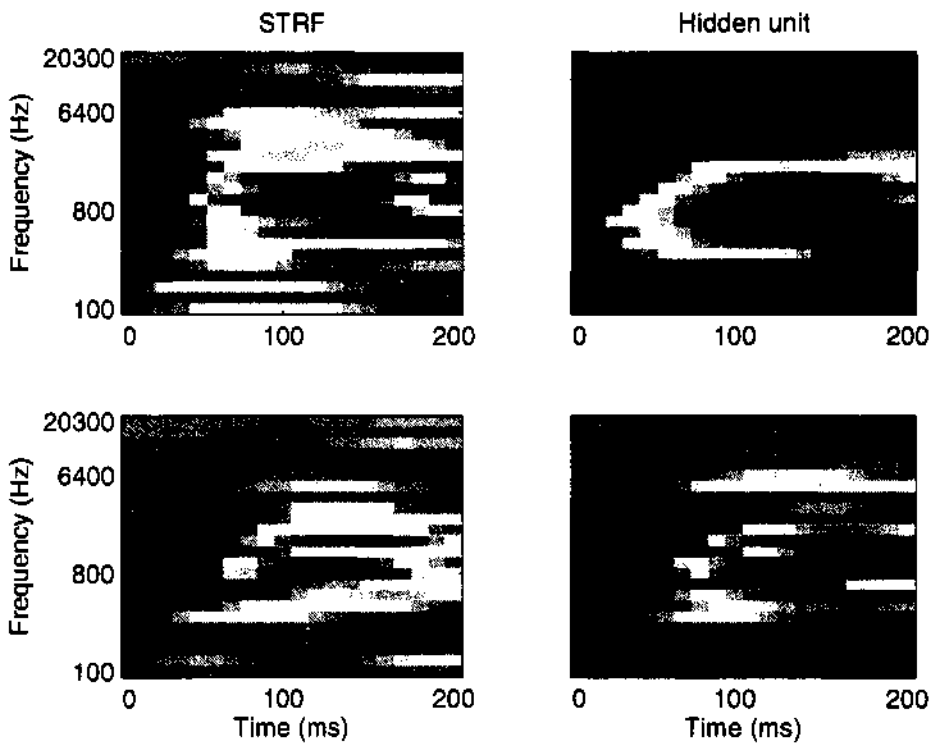

Figure 4.9: Neural networks recovered the linear components of neuronal stimulus-response transformations. Left panels show STRFs recovered from two neurons. Right panels show interpretation of weights of single hidden units recovered from neural networks trained for the same neurons. To obtain the single hidden unit we first identified the training cycle with the best neural network, and then selected the network with one hidden unit from that cycle.

Chapter 5

Discussion

We used artificial neural networks as nonlinear approximators to study stimulus-response transformation in single auditory neurons probed with an ensemble of natural sounds. Neural networks—on average—outperformed linear models of neuronal transformations and were able to predict more details of neuronal responses. In addition, neural networks were able to recover linear components of neuronal transformation similar to pure linear models (STRFs), confirming that the networks enhanced the linear models by performing multidimensional nonlinear regression.

Neural networks have been previously used to study neuronal transformations in the visual cortex (Lau et al., 2002; Lehky and Sejnowski, 1990; Lehky et al., 1992; Prenger et al., 2004). To our knowledge, this study represents the first attempt to study nonlinearity of neuronal responses using neural networks in the auditory cortex. One other study in the auditory system (Bankes and Margoliash, 1993) used similar methodology to study properties of neurons in auditory thalamus (*i.e.*a subcortical nucleus) of zebra finches. This study, however, differs from all previous study in that we have used subthreshold voltage fluctuations as a measure of neuronal response. This enabled us to remove the final nonlinearity in neuronal transformation, namely the all-or-nothing generation of action potentials, and effectively use the total neuronal input as a measure of single neuron computation, *i.e.*the stimulus-response transformation.

The prediction capability of our neural network models is comparable with studies in the visual system, with correlation between predicted and actual responses ranging from 0.78 (Lehky et al., 1992); approximately 0.40 (Lau et al., 2002); to 0.24 (Prenger et al., 2004). This is rather surprising because much more is known

about different neuronal classes and properties in the visual system, whereas the auditory physiology lags behind. Thus, even without any assumptions about the class of cells we recorded from, we (the neural networks) were able to achieve a comparable performance. Bankes and Margoliash (Bankes and Margoliash, 1993) reported higher prediction correlations for single cells. Their study, however, was conducted in subcortical nucleus where neuronal properties are more linear than properties of cortical neurons.

The seemingly low prediction capability of neural networks (mean correlation coefficient of 0.50) in this study can be explained by several factors. First, non-stationarity and variability of neuronal responses can influence prediction ability of neuronal networks. For most cells we used several trials to compute the mean response, but the variability of responses. Second, the discretization of stimuli and response can also affect network performance, because the choice of time windows can decrease response noise by averaging on longer time scales, or multiple repeats. Previous studies (Lau et al., 2002; Lehky et al., 1992; Prenger et al., 2004) used longer time bins (14–160 ms) and, in some cases, smoothing of response data.

Third, we used an ensemble of natural sounds which contained a rich set of spectral and temporal correlations. It is, however, possible that even these stimuli did not cover the important parts of the highly dimensional space of all acoustic features. Thus, the best combination of training and validation datasets seems to be large training sets collected with variable set of stimuli (perhaps with single repeats), together with repeated presentations of validation data set (perhaps with less variable stimuli).

And last, but no least, the in-vivo patch-clamp whole-cell recordings we used have different selection bias from all previous studies. The conventional recoding techniques select cells based on their activity and responsiveness, whereas patch-clamp techniques select cells based experimentalist's ability to make a gigaohm seal between the electrode and neuronal membrane. Thus our set of neurons likely included cells with very different properties.

Neural network models do not make any assumptions about the underlying properties of stimuli, or neuronal transformations. The neural network methodology can recover arbitrary nonlinear transformations, and—as these methods proved to be useful in the visual system (Lau et al., 2002; Lehky et al., 1992; Prenger et al., 2004)—further characterization of these nonlinearities in the auditory system represents an exciting challenge for further research.

List of Figures

Bibliography

Aertsen, A. M. and Johannesma, P. I. (1981). The spectro-temporal receptive field. a functional characteristic of auditory neurons. *Biol Cybern* **42**:133–143.

Bankes, S. C. and Margoliash, D. (1993). Parametric modeling of the temporal dynamics of neuronal responses using connectionist architectures. *J Neurophysiol* **69**:980–991.

Barlow, H. B. (1972). Single units and sensation: a neuron doctrine for perceptual psychology? *Perception* **1**:371–394.

Blake, D. T. and Merzenich, M. M. (2002). Changes of AI receptive fields with sound density. *J Neurophysiol* **88**:3409–3420.

Brodmann, K. (1909). *Vergleichende Lokalisationslehre der Großhirnrinde in ihren Prinzipien dargestellt auf Grund des Zellenbaues*. Barth JA, Leipzig.

Dayan, P. and Abbott, L., editors (2001). *Theoretical Neuroscience. Computational and Mathematical Modeling of Neural Systems*. Computational Neuroscience. MIT Press, Cambridge, MA 1st edition.

deCharms, R., Blake, D., and Merzenich, M. (1998). Optimizing sound features for cortical neurons. *Science* **280**:1439–1443.

Demuth, H. and Beale, M. (2005). *Neural Network Toolbox, User's Guide, Version 4*. The Mathworks, Inc., Natick, MA, revised for version 4.0.5.

DeWeese, M. R., Hromádka, T., and Zador, A. M. (2005). Reliability and representational bandwidth in the auditory cortex. *Neuron* **48**:479–488.

Duda, R. O., Hart, P. E., and Stork, D. G. (2004). *Pattern Classification*. John Wiley & Sons 2nd edition.

Ehret, G. (1997). The auditory cortex. *J Comp Physiol [A]* **181**:547–557.

Einhäuser, W., Kayser, C., König, P., and Körding, K. P. (2002). Learning the invariance properties of complex cells from their responses to natural stimuli. *Eur J Neurosci* **15**:475–486.

Field, D. J. (1987). Relations between the statistics of natural images and the response properties of cortical cells. *J Opt Soc Am A* **4**:2379–2394.

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning. Data mining, inference and prediction.* Springer Series in Statistics. Springer.

Haykin, S. (1999). *Neural Networks. A Comprehensive Foundation.* Prentice-Hall Inc. 2nd edition.

Hebb, D. (1949). *The Organization of Behavior.* Wiley.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proc Natl Acad Sci U S A* **79**:2554–2558.

Hromádka, T., DeWeese, M., and Zador, A. Sparse representation of sounds in the unanesthetized auditory cortex. submitted.

Hubel, D. H. and Wiesel, T. N. (1977). Ferrier lecture. Functional architecture of macaque monkey visual cortex. *Proc R Soc Lond B Biol Sci* **198**:1–59.

Kandel, E. R., Schwartz, J. H., and Jessell, T. M. (2000). *Principles of Neural Science.* McGraw-Hill/Appleton & Lange 4th edition.

Klein, D. J., Depireux, D. A., Simon, J. Z., and Shamma, S. A. (2000). Robust spectrotemporal reverse correlation for the auditory system: optimizing stimulus design. *J Comput Neurosci* **9**:85–111.

Kolmogorov, A. N. (1957). On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition. *Doklady Akademii Nauk SSSR* **114**:953–956.

Kowalski, N., Depireux, D. A., and Shamma, S. A. (1996). Analysis of dynamic spectra in ferret primary auditory cortex. I. Characteristics of single-unit responses to moving ripple spectra. *J Neurophysiol* **76**:3503–3523.

Kůrková, V. (1991). Kolmogorov's theorem is relevant. *Neural Computation* **3**:617–622.

Kůrková, V. (1992). Kolmogorov's theorem and multilayer neural networks. *Neural Computation* **5**:501–506.

Kůrková, V. (2002). Neural networks as universal approximators. In Arbib, M. A., editor, *The Handbook of Brain Theory and Neural Networks* pages 1180–1183. MIT Press, Cambridge 2nd edition.

Lau, B., Stanley, G. B., and Dan, Y. (2002). Computational subunits of visual cortical neurons revealed by artificial neural networks. *Proc Natl Acad Sci U S A* **99**:8974–8979.

Lehky, S. R. and Sejnowski, T. J. (1990). Neural network model of visual cortex for determining surface curvature from images of shaded surfaces. *Proc R Soc Lond B Biol Sci* **240**:251–278.

Lehky, S. R., Sejnowski, T. J., and Desimone, R. (1992). Predicting responses of nonlinear neurons in monkey striate cortex to complex patterns. *J Neurosci* **12**:3568–3581.

Linden, J. F., Liu, R. C., Sahani, M., Schreiner, C. E., and Merzenich, M. M. (2003). Spectrotemporal structure of receptive fields in areas AI and AAF of mouse auditory cortex. *J Neurophysiol* **90**:2660–2675.

Maass, W. and Bishop, C. M., editors (2001). *Pulsed Neural Networks*. The MIT Press.

Machens, C., Wehr, M., and Zador, A. (2004). Linearity of cortical receptive fields measured with natural sounds. *J Neurosci* **24**:1089–1100.

McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics* **5**:115–133.

Minsky, M. and Papert, S. (1969). *Perceptrons*. MIT Press, Cambridge, MA.

Møller, M. F. (1993). A scaled conjugate gradient for fast supervised learning. *Neural Networks* **6**:525–533.

Moshitch, D., Las, L., Ulanovsky, N., Bar-Yosef, O., and Nelken, I. (2006). Responses of neurons in primary auditory cortex (A1) to pure tones in the halothane-anesthetized cat. *J Neurophysiol.*

Nelken, I., Prut, Y., Vaadia, E., and Abeles, M. (1994). Population responses to multifrequency sounds in the cat auditory cortex: one- and two-parameter families of sounds. *Hear Res* **72**:206–222.

Nelken, I., Rotman, Y., and Yosef, O. B. (1999). Responses of auditory-cortex neurons to structural features of natural sounds. *Nature* **397**:154–157.

Nguyen, D. and Widrow, B. (1990). Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. *Proceedings of the International Joint Conference on Neural Networks* **3**:21–26.

Nicholls, J. G., Martin, A. R., Wallace, B. G., and Fuchs, P. A. (2001). *From Neuron to Brain*. Sinauer Associates, Inc 4th edition.

Pavlásek, J. and Hromádka, T. (2001). Neural network comparing two-rate-encoded inputs entering in parallel. *Gen Physiol Biophys* **20**:61–82.

Pinkus, A. (1998). Approximation theory of the mpl model in neural networks. *Acta Numerica* **8**:277–283.

Poirazi, P., Brannon, T., and Mel, B. W. (2003a). Arithmetic of subthreshold synaptic summation in a model CA1 pyramidal cell. *Neuron* **37**:977–987.

Poirazi, P., Brannon, T., and Mel, B. W. (2003b). Pyramidal neuron as two-layer neural network. *Neuron* **37**:989–999.

Prenger, R., Wu, M. C.-K., David, S. V., and Gallant, J. L. (2004). Nonlinear V1 responses to natural scenes revealed by neural network analysis. *Neural Netw* **17**:663–679.

Press, W., SA, T., WT, V., and BP, F. (1992). *Numerical recipes in C*. Cambridge University Press 2nd edition.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychol Rev* **65**:386–408.

Rotman, Y., Bar-Yosef, O., and Nelken, I. (2001). Relating cluster and population responses to natural sounds and tonal stimuli in cat primary auditory cortex. *Hear Res* **152**:110–127.

Rumelhart, D. E., Hinton, G. E., and Wiliams, R. J. (1986). Learning representations by back-propagating errors. *Nature* **323**:533–536.

Sally, S. L. and Kelly, J. B. (1988). Organization of auditory cortex in the albino rat: sound frequency. *J Neurophysiol* **59**:1627–1638.

Sen, K., Theunissen, F. E., and Doupe, A. J. (2001). Feature analysis of natural sounds in the songbird auditory forebrain. *J Neurophysiol* **86**:1445–1458.

Shepherd, G. M., editor (2004). *The Synaptic Organization of the Brain*. Oxford University Press 5th edition.

Šíma, J. and Neruda, R. (1996). *Teoretické otázky neuronových sítí.* Matfyzpress, Praha 1st edition.

Stevens, C. F. and Zador, A. M. (1998). Input synchrony and the irregular firing of cortical neurons. *Nat Neurosci* **1**:210–217.

Theunissen, F. E., Woolley, S. M. N., Hsu, A., and Fremouw, T. (2004). Methods for the analysis of auditory processing in the brain. *Ann N Y Acad Sci* **1016**:187–207.

Tian, B., Reser, D., Durham, A., Kustov, A., and Rauschecker, J. P. (2001). Functional specialization in rhesus monkey auditory cortex. *Science* **292**:290–293.

Widrow, B. (1962). Generalization and information storage in networks of adeline 'neurons'. In Yovitz, M., Jacobi, G., and Goldstein, G., editors, *Self-Organizing Systems* pages 435–461. Spartan Books, Washington, DC.